

A Thesis Submitted for the Degree of PhD at the University of Warwick

Permanent WRAP URL:

<http://wrap.warwick.ac.uk/109864>

Copyright and reuse:

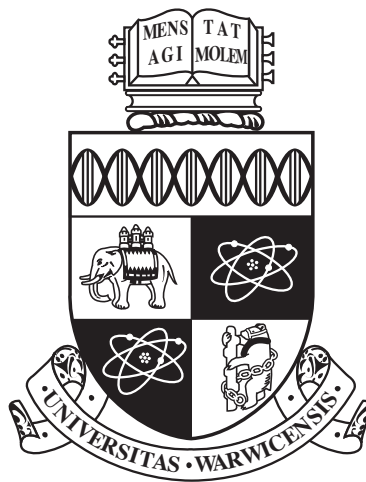
This thesis is made available online and is protected by original copyright.

Please scroll down to view the document itself.

Please refer to the repository record for this item for information to help you to cite it.

Our policy information is available from the repository home page.

For more information, please contact the WRAP Team at: wrap@warwick.ac.uk



Evolutionary Multi-objective Worst-case Robust Optimisation

by

Ke Lu

Thesis

Submitted to the University of Warwick

for the degree of

Doctor of Philosophy

.....

September 2017

THE UNIVERSITY OF
WARWICK

Contents

Acknowledgments	iv
Declarations	v
Abstract	vi
Chapter 1 Introduction	1
1.1 Motivation	1
1.2 Research objectives	3
1.3 Thesis outline	3
Chapter 2 Background and literature review	5
2.1 Multi-Objective Optimisation Problems	5
2.2 Robust solutions	6
2.3 Problem definition	9
2.4 Evolutionary algorithms	10
2.5 Test functions	12
2.5.1 Test function 1	12
2.5.2 Test function 2	14
2.5.3 Test function 3	14
2.6 Performance measure	15
2.7 A review of related work	16
2.7.1 Single-objective robust optimisation	17
2.7.2 Multi-objective robust optimisation	18
2.7.3 Reliability	21
2.7.4 Active robustness	22
Chapter 3 Finding the trade-off between worst-case quality and robustness	23

3.1	Motivation of the developed algorithms	23
3.2	Proposed solution approach	26
3.2.1	The point-based nested MOEA	27
3.2.2	The envelope-based nested MOEA	28
3.3	Empirical evaluation	35
3.3.1	Parameter settings	35
3.3.2	Test results	36
3.4	Summary	41

Chapter 4 Improving the efficiency of bi-level worst-case optimisation 42

4.1	The link to bi-level optimisation	42
4.2	Background introduction	43
4.3	Algorithm and New Strategies for point-based algorithm	44
4.3.1	Worst case bi-level evolutionary algorithm	44
4.3.2	Strategies to save fitness evaluations for point-based algorithm	45
4.4	Empirical Results for point-based algorithm	48
4.4.1	Parameter settings	48
4.4.2	Test results and analysis	49
4.5	Test results for a given number of fitness evaluations	51
4.6	Test results for the combination of three strategies	54
4.7	Summary	55

Chapter 5 Improving the efficiency of envelope-based algorithm 56

5.1	Motivation	56
5.2	Strategies to save fitness evaluations for envelope-based algorithm .	57
5.2.1	Strategy I: Exploit upper level information for early abortion of the lower level	58
5.2.2	Strategy II: Exploit the information in the neighbourhood . .	60
5.2.3	Strategy III: Skip re-evaluation if there is no improvement of the lower level front	63
5.2.4	Strategy IV: Lower level smart initialisation	63
5.2.5	Strategy V: Lower level generations adaptive to Δ	64
5.2.6	Strategy VI: Lower level population size adaptive to Δ . . .	64
5.3	Empirical results for envelope-based algorithm by applying simple strategies	64
5.3.1	Parameter settings	65
5.3.2	Test results and analysis	65

5.4	Summary	73
Chapter 6	Surrogate-assisted algorithms	75
6.1	Background and related work	77
6.2	Gaussian processes	78
6.3	Surrogate-assisted point-based algorithm	79
6.3.1	Surrogate-assisted point-based algorithm framework	79
6.3.2	Re-evaluation in surrogate-assisted point-based algorithm	80
6.3.3	Identity promising upper level solutions	81
6.3.4	Building Gaussian process model	82
6.4	Surrogate-assisted envelope-based algorithm	83
6.4.1	Surrogate-Assisted envelope-based algorithm framework	84
6.4.2	Re-evaluation in surrogate-assisted envelope-based algorithm	85
6.4.3	Identify “good” upper level solutions	86
6.4.4	Gaussian Process model for envelope-based algorithm	87
6.5	Empirical evaluation and analysis	89
6.5.1	Empirical evaluation of surrogate-assisted point-based algorithm	89
6.5.2	Empirical evaluation of surrogate-assisted envelope-based algorithm	93
6.6	Summary	94
Chapter 7	Conclusion and future work	97
7.1	Research summary	97
7.2	Contributions	98
7.3	Limitations	101
7.4	Future research work	102
	Bibliography	103

Acknowledgments

I would like to express my foremost appreciation to Professor Juergen Branke for his continuous support, great patience, wise advice, encouragement throughout my PhD study and related research at Warwick Business School. I could not have imagined having a better supervisor to supervise my PhD study. His invaluable insights and guidance helped me accomplish the thesis.

I would also like to take this opportunity to thank Professor Tapabrata Ray, for all his valuable comments and suggestions. I also thank my second supervisor Nalan Gulpinar for her encouragement.

Last but not least, I would like to thank my family. Especially my mother who always supports me and encourages me over my PhD study. In my first year of the PhD study, I was depressed and everything seemed hopeless. It was my mother who encouraged me to continue my study. Because of their unconditional support, I have the determination to finish my study.

Declarations

This thesis is submitted to the University of Warwick in support of my application for the degree of Doctor of Philosophy. It has been composed by myself and has not been submitted in any previous application for any degree. The work presented was carried out by the author.

Parts of this thesis have been published by the author:

- Juergen Branke and Ke Lu. Finding the trade-off between robustness and worst-case quality. In Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation, pages 623-630. ACM, 2015.
- Ke Lu, Juergen Branke, and Tapabrata Ray. Improving efficiency of bi-level worst case optimization. In International Conference on Parallel Problem Solving from Nature, pages 410-420. Springer, 2016.

Abstract

Many real-world problems are subject to uncertainty, and often solutions should not only be good, but also robust against environmental disturbances or deviations from the decision variables. While most papers dealing with robustness aim at finding solutions with a high *expected* performance given a distribution of the uncertainty, we examine the trade-off between the allowed deviations from the decision variables (tolerance level), and the worst case performance given the allowed deviations. In this research work, we suggest two multi-objective evolutionary algorithms to compute the available trade-offs between allowed tolerance level and worst-case quality of the solutions, and the tolerance level is defined as robustness which could also be the variations from parameters. Both algorithms are 2-level nested algorithms. While the first algorithm is point-based in the sense that the lower level computes a point of worst case for each upper level solution, the second algorithm is envelope-based, in the sense that the lower level computes a whole trade-off curve between worst-case fitness and tolerance level for each upper level solution.

Our problem can be considered as a special case of bi-level optimisation, which is computationally expensive, because each upper level solution is evaluated by calling a lower level optimiser. We propose and compare several strategies to improve the efficiency of both algorithms. Later, we also suggest surrogate-assisted algorithms to accelerate both algorithms.

Chapter 1

Introduction

1.1 Motivation

Many real-world optimisation problems are subject to uncertainties that are practically impossible to avoid. For instance, in manufacturing, it is usually impossible to produce an item exactly following the design specifications as shown Figure 1.1, there are always some manufacturing tolerances. The solutions could be potentially risky to use if uncertainties have not been taken into account during optimisation. Hence, when solving real-world optimisation problems, it is important to consider solutions that are not only globally optimal but also practical to use in reality despite the different uncertainties present within those problems. In other words, people prefer solutions that not only have good quality but also have tolerance against uncertainties.

One extensively used concept of robustness for single-objective optimisation is proposed by Branke (1998). The objective function is replaced by its mean function, for any solution x , it maps to its average function value in a pre-defined neighbourhood of x . The mean function is minimised instead of the objective function. Those

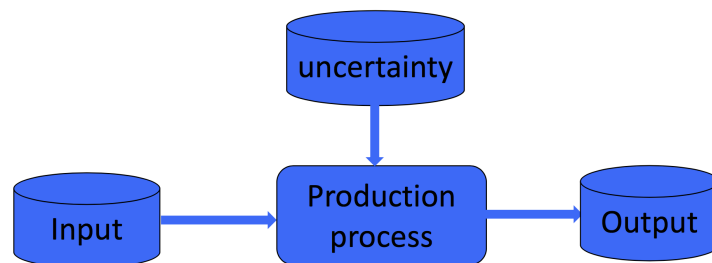


Figure 1.1: An example in manufacturing

solutions whose function values in the neighbourhood does not change much are considered as good.

Two concepts of robustness for multi-objective optimisation are introduced in Deb & Gupta (2006). The first one replaces the objective functions by the mean functions as Branke (1998) for a single objective function. Robust solutions are defined as the efficient solutions obtained by optimising the mean functions. The second concept optimises the original objective functions with the constraints that does not allows the variations between the objective function values and the mean function values to be greater than a certain limit. The second concept gives the users an option to define the level of robustness. Our approach also allows the user to make decisions with different robustness levels. The difference is that we look at the worst-case fitness, and get a trade-off between worst-case fitness and robustness. In the second concept, the optimal solutions are based on a single pre-defined robustness level, for different robustness levels, the problem is solved more times.

The first concept in Deb & Gupta (2006) is extended in Barrico & Antunes (2006) by introducing the degree of robustness if a solution. A feasible solution has a predefined neighbourhood, and it measures how much this neighbourhood can be extended with the constraints that the variations between the objective functions values and the mean function values can not be larger than a predefined limit.

Gunawan & Azarm (2005) introduce sensitivity region with uncertainty in the parameters space to measure the robustness in multi-objective optimisation problems. The allowed variations of the parameters is defined as the sensitivity region by restricting the variations of the objective function values within a certain limit. They also introduce worst-case-sensitivity region because the sensitivity region could not be asymmetric. This worst-case region fits into the sensitivity region with a maximum radius.

Avigad & Branke (2008) propose an evolutionary algorithm to search for solutions in a multi-objective optimisation problem with uncertainty in the design and parameter space. Each nominal solution corresponds to a worst set of scenarios. The algorithm aims to search for solutions that have the best worst-case performance.

In this thesis, we will look at the uncertainty in decision space. On the one hand, we look at the quality of a solution. On the other hand, its good tolerance against uncertainty will be desired. The aim is to deal with the uncertainty and search for robust solutions for a given tolerance level. There will be more than one criterion to consider (solution quality and good tolerance against uncertainty).

Risk averse decision makers may care about the worst-case performance with decision variables disturbed by uncertainty. If the worst-case situation involves potential bankruptcy, death or system breakdown, it is of great importance to look at the worst-case quality. For a solution with a certain tolerance level, the worst-case performance will be considered. A possible application are manufacturing tolerances, where an engineer can specify an allowed tolerance for manufacturing. A low tolerance requirement incurs substantially higher manufacturing cost, whereas a high tolerance usually means having to accept a lower worst-case quality of the solution. We address the uncertainty in the decision space, and a solution with good worst-case performance is considered as robust against uncertainty. We attempt to search for the trade-off between worst-case quality and tolerance levels. The trade-off would provide information to decision makers so that they can make informed decisions with respect to personal preferences.

1.2 Research objectives

For an optimisation problem with uncertainty in the decision space, solutions that not only have good quality but also have good robustness are preferred. We will look at how uncertainty is modelled within the problems and define robust solutions and robustness. In this research, we consider the worst-case performance if disturbed by uncertainty in the decision space. We would like to study a trade-off between the worst-case performance and robustness. This research seeks to achieve the following objectives.

1. Propose a formal framework which defines the formulations of the worst-case performance and robustness.
2. Develop algorithms to search for the trade-off between worst-case quality and robustness.
3. Improve the efficiency of the developed algorithms.

1.3 Thesis outline

The overall structure of the thesis takes the form of seven chapters.

Chapter 2 provides a brief description of the multi-objective optimisation problem and defines the problem framework of this research. For multi-objective optimisation problems, the optimal solutions are a set of non-dominated solutions with different trade-offs. Because we investigate uncertainty in the decision space,

the robust solutions are described. Robustness in our thesis is defined as the maximal allowed deviation from the decision variables. The problem we solve is explained and formulated in the problem definition. What follows is a set of test functions that are used to test our algorithms and methods proposed in the following chapters. The performance measure Inverted Generational Distance is applied to evaluate the quality of the Pareto optimal solutions obtained by each algorithm. The final section discusses related work in robust optimisation.

In Chapter 3, two newly developed algorithms are suggested to solve the problem described in Chapter 2. The first algorithm is point-based in the sense that the lower level is a single objective optimisation problem that returns a single value to the upper level. The second algorithm is envelope-based because the lower level is a multi-objective optimisation problem which returns a set of solutions with different trade-offs (envelope) to the upper level.

Both algorithms in Chapter 3 are bi-level which is computationally expensive, because each upper level solution is evaluated by calling a lower level optimiser that, in our case, is population based. Therefore, in Chapter 4, we propose and compare several strategies to reduce the computational costs of the point-based algorithm. The aim of this chapter is to obtain a reasonably good Pareto front at upper level given a small number of fitness evaluations. Chapter 5 extends the strategies to the envelope-based algorithm.

In Chapter 6, we combine our algorithms with surrogate models in order to accelerate the algorithms. Generally, a surrogate model is built to approximate the actual objective function values. In our case, we would like to apply surrogate models to learn the robust function rather than actual fitness. The surrogate model is used to identify those solutions are promising in the upper level, and only the good solutions are evaluated by running the lower level algorithm.

Chapter 7 concludes the thesis with a summary of the contributions and limitations, as well as some ideas for future work.

Chapter 2

Background and literature review

Considering decision variables disturbed by uncertainty, we often prefer solutions that are not only good, but also robust against environmental disturbances or deviations from the decision variables. While most papers dealing with uncertainty in the decision space aim to find solutions with a high *expected* performance given a distribution of the uncertainty, we examine the trade-off between the allowed deviations from the decision variables (tolerance level), and the worst case performance given the allowed deviations.

2.1 Multi-Objective Optimisation Problems

Assuming maximisation without loss of generality, the multi-objective optimisation problems (MOPs) could be defined as follows:

$$\max \quad f(x) = f_1(x), f_2(x), \dots, f_m(x) \quad (2.1)$$

s. t.

$$g(x) \leq 0 \quad (2.2)$$

$$h(x) = 0 \quad (2.3)$$

where $f(x)$ is the objective function vector to be optimised that consists of m objectives, and $g(x)$ and $h(x)$ are the inequality constraints and equality constraints, respectively.

For the single objective optimisation problems, the global optimal objective

value is uniquely defined. However, when there is more than one objectives to be optimised, the optimal solutions are likely to consist of a set of alternative trade-offs. We now define the basic terminology as follows.

- **Pareto Dominance**

Without loss of generality, for a multi-objective maximisation problem, the Pareto dominance relationship can be described as follows:

A solution vector x is said to dominate another y if $f_i(x) \geq f_i(y)$ for all objectives, and $f_i(x) > f_i(y)$ for at least one i .

- **Pareto Optimal Solution Set**

A solution x to the problem is defined as Pareto optimal if it is not dominated by any other feasible solution y .

- **Pareto Front**

The image of all Pareto optimal solutions in the objective space is known as Pareto front, and it is also known as efficient front. Figure 2.1 shows the dominance relationship for a two objectives maximisation problem, where f_1 and f_2 are the two objectives to be maximised. Solutions A-D are non-dominated solutions that form the Pareto front. Solution E is dominated by A and B, and Solution F is dominated by C. Solution G is dominated by all other solutions.

2.2 Robust solutions

In this section, we describe robust solutions with uncertainty in decision space and introduce the robustness definition.

1. Uncertainty in decision space

With uncertainty in the decision space, the objective function can be expressed as:

$$f(x + \delta) \tag{2.4}$$

where δ is the uncertainty vector in decision variables. Usually δ is restricted within a specified range or follows a known distribution.

2. Robustness

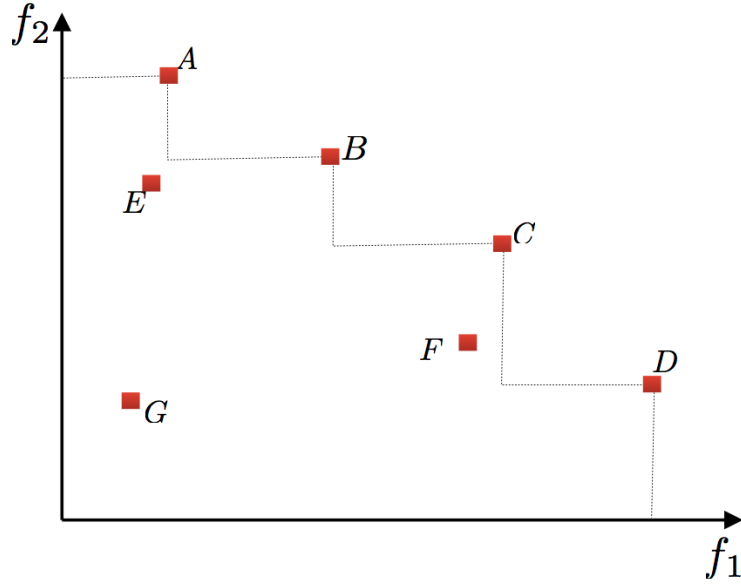


Figure 2.1: An example of Pareto dominance relationship between two objectives to be maximised

There are several robustness definitions. If a solution has tolerance against uncertainty in the decision space, we say this solution is robust and has good tolerance. In our research, the robustness is defined as the allowed deviation from the decision variables which is also called as tolerance level.

3. Robust solutions

Typically, evolutionary algorithms aim to find a globally optimal solution. However, if such a solution is very sensitive to small variations of the decision variables or operating environment, in practice, people may prefer solutions that have perhaps slightly inferior quality but also have good *tolerance* to uncertainty. There are a number of different definitions for robust solutions (Branke 2001), including having a good expected performance and a good worst-case performance. Generally, a robust solution is defined as a solution whose fitness is not sensitive to small variations in decision space or environment. If a solution has good tolerance to small variations of decision variables, we say this solution has good tolerance against uncertainty.

From Figure 2.2, if two solutions A and B are disturbed by uncertainty δ that is bounded to $[-\theta, \theta]$, the solution quality of B in terms of f varies much more than A . Solution A is less sensitive to uncertainty and more robust than B .

If we consider the expected value of the objective function within the distur-

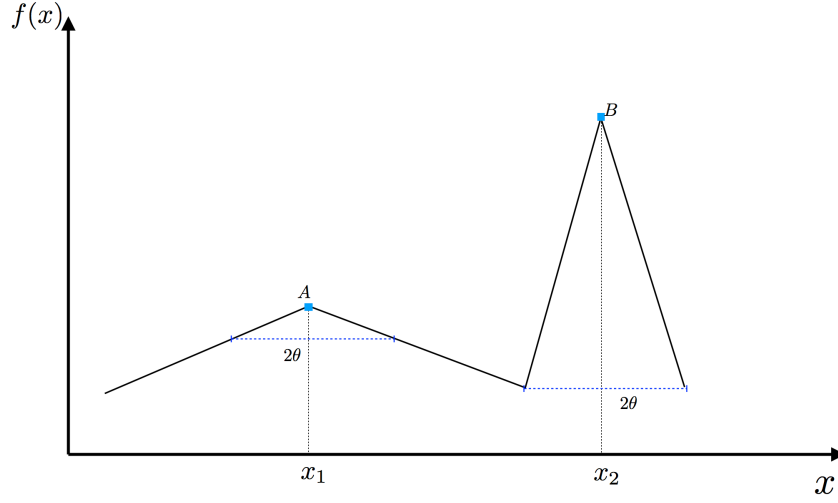


Figure 2.2: An example function used for description of robust solutions

bance region,

$$f^E(x) = \int p(\delta) f(x + \delta) d\delta$$

where $p(\delta)$ is the probability density function of δ .

We would like to maximise the expected value

$$\max_x f^E(x)$$

If we look at the worst-case value in the disturbance region,

$$f^w(x) = \min_{\delta} f(x + \delta)$$

where $\delta \in [-\Delta, \Delta]$, and Δ is the maximal disturbance for x .

We would like to maximise the worst-case value

$$\max_x f^w(x)$$

In our research, we consider the worst-case quality in the disturbance region. Usually, for a solution as its tolerance level increases, the worst-case quality within the disturbance region will drop. Solutions with greater tolerance level and better worst-case fitness are preferred. As shown in Figure 2.2, the decision variable may be disturbed by δ belongs to the range $[-\theta, \theta]$, solution A has

better worst-case quality than B .

2.3 Problem definition

We assume that an objective function $f(x)$ and an allowable range for $x \in [x^{\min}, x^{\max}]$ are given. The optimisation is not only over x , but the user can also set a tolerance level δ , and the goal is to identify the trade-off between tolerance level δ and worst case performance within the tolerance region $[x - \delta, x + \delta]$, which we assume to be symmetric around x . Without loss of generality assuming a maximisation problem here, the two objectives we want to maximise are $f^w(x, \delta) = \min_y \{f(y) | y \in [x - \delta, x + \delta]\}$ and δ .

To summarise, the aim is to identify all Pareto optimal solutions for the following 2-objective optimisation problem:

$$\max \quad f^w(x, \delta) \quad (2.5)$$

$$\max \quad \delta \quad (2.6)$$

s. t.

$$x \in [x^{\min}, x^{\max}] \quad (2.7)$$

$$\delta \in [0, \min\{x - x^{\min}, x^{\max} - x\}] \quad (2.8)$$

Equation 2.8 ensures that the tolerance region can not exceed the allowed search space (i.e., a solution at the border of the feasible space can not have a tolerance lever greater than zero). Note that this is a nested optimisation problem, as the calculation of $f^w(x, \delta) = \min_y \{f(y) | y \in [x - \delta, x + \delta]\}$ is itself an optimisation problem.

Because the space for x is bounded, the maximal possible tolerance level of a solution x is bounded by how far away x is from the boundary. As a result, the feasible space is a triangle and is shown in Figure 2.3 for a one dimensional problem.

For a univariate problem, it can be represented by the framework described above straightforward. We give the representation of a two-dimensional problem $f(x_1, x_2)$ with x_1 and x_2 , $x = (x_1, x_2)$, where

$$x_1 \in [x_1^{\min}, x_1^{\max}]$$

$$x_2 \in [x_2^{\min}, x_2^{\max}]$$

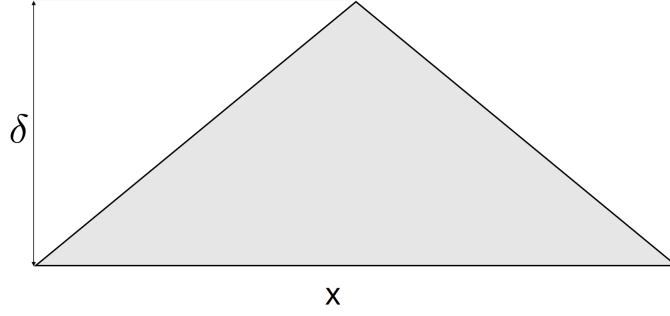


Figure 2.3: The feasible search space

For these two variables, the robustness for each variable is restricted to

$$\delta_1 \in [0, \min\{x_1 - x_1^{\min}, x_1^{\max} - x_1\}]$$

$$\delta_2 \in [0, \min\{x_2 - x_2^{\min}, x_2^{\max} - x_2\}]$$

We can get that the robustness δ is bounded to $[0, \min\{\delta_1, \delta_2\}]$. The worst-case fitness calculation is displayed as

$$f^w(x_1, x_2, \delta) = \min_{y_1, y_2} \{f(y_1, y_2) | y_1 \in [x_1 - \delta, x_1 + \delta], y_2 \in [x_2 - \delta, x_2 + \delta]\}$$

The problem is formulated as a multi-objective optimisation problem where the two objectives worst-case quality and robustness are optimised. The optimal solutions to this problem will be a set of non-dominated solutions. We would like to develop algorithms to search for the trade-off between worst-case quality and robustness. This will provide information to help the decision maker. For a certain robustness, the best worst-case performance can be identified from the trade-off.

Evolutionary Algorithms (EAs) are able to search for a set of near-optimal solutions with different trade-offs at the end of an optimisation procedure. The main techniques applied in this research are based on multi-objective EAs.

2.4 Evolutionary algorithms

EAs have been extensively used to deal with complex optimisation problems (Coello et al. 2007). EAs are stochastic optimisation methods derived from natural evolution: given a randomly initialised population of individuals in the search space, these individuals evolve with respect to the Darwinian principle of the survival of the fittest. The fitness of the population increases with the evolution. Without

loss of generality, given an objective function to be maximised, randomly generate a set of candidate solutions, each solution is evaluated in terms of its fitness. The fitness function measures how well the solutions perform. The solution with higher fitness is considered better. Based on the fitness evaluations, some of the solutions in the population are selected to survive to the next generation. The probability of survival of the new individual depends on their fitness: the individuals with a high fitness are more likely to be kept while individuals with low fitness would be discarded.

Three major evolutionary operators are mutation, recombination and/or selection, and they are applied to generate new solutions. Recombination is used to produce one or more new solutions by selection two or more solutions in the population. Mutation is applied to one solution to create a new solution. The offspring population that is the newly generated solutions by recombination and mutation. Each solution in the offspring population is evaluated based on its fitness. The solutions in both population and offspring compete in terms of the fitness, and select a number of better solutions that survive to the next generation. This describes one iteration of the evolution, and this process evolves until the optimal solutions are obtained or the stopping condition is satisfied.

The flowchart of EAs could be expressed as Figure 2.4, which describes the process of EAs (Eiben et al. 2003). They summarise the evolutionary operators as follow:

- Variation operators that contains recombination and mutation, and they are applied to create new solutions in the population.
- Selection operator drives the improvement of the solutions quality in the population.

In some variants of evolutionary algorithms selection is deterministic, always selecting the best individual. While the selection could be probabilistic, and the best individuals are not selected deterministically.

The general evolutionary algorithm is described as initially generate a set of candidate solutions randomly that form the parent population, and evaluate each solution in the parent population. Repeat the following process until the stopping condition is satisfied. Select solutions from the parent population, recombine the selected solutions to generate offsprings and mutate the offsprings to get the offspring population. Evaluate each new solution in the offspring population, select good individuals from the combined parent and offspring population that goes to the next generation. A population whose quality is getting better as the EA evolves.

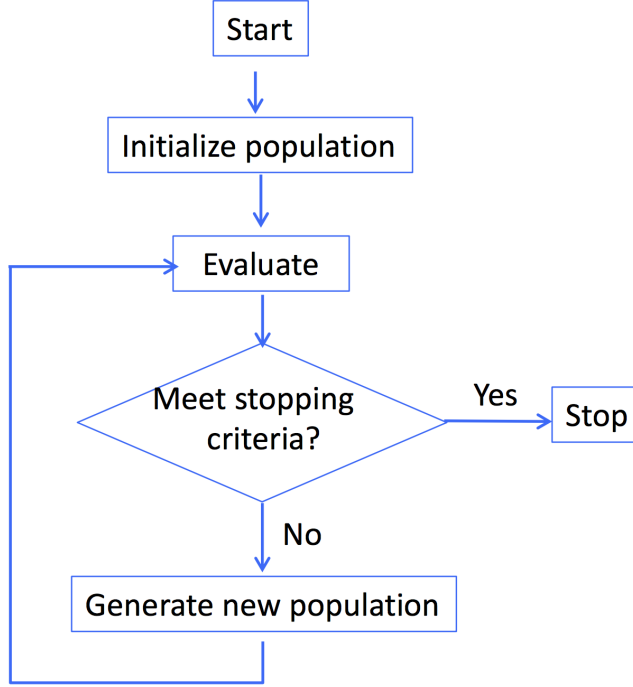


Figure 2.4: The flow chart of a standard EA

One of the main advantages of evolutionary algorithms is that all they need is the objective function and fitness measures about the optimisation problem. They can deal with constrained or unconstrained non-linear problems that are defined in discrete or continuous search spaces. Besides, the evolutionary algorithms search for the optimal solutions in the whole search space and this makes them global.

2.5 Test functions

2.5.1 Test function 1

We propose the following simple one-dimensional function $f(x)$, which is visualised in Figure 2.5:

$$f(x) = \begin{cases} \frac{5}{3}x - \frac{2}{3} & \text{if } 1 < x \leq \frac{5}{2} \\ -\frac{5}{3}x + \frac{23}{3} & \text{if } \frac{5}{2} \leq x \leq 4 \\ \frac{2}{3}x - \frac{2}{3} & \text{if } 4 < x \leq \frac{11}{2} \\ -\frac{2}{3}x + \frac{20}{3} & \text{if } \frac{11}{2} < x \leq 7 \end{cases}$$

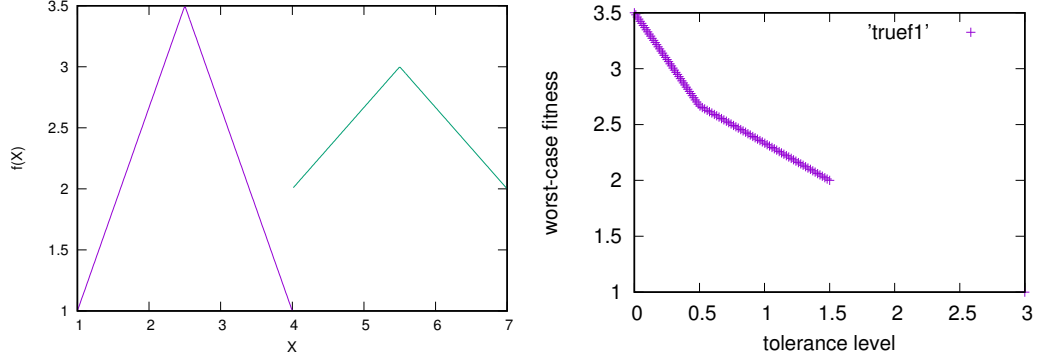


Figure 2.5: The plots of test function 1 (left) and its true Pareto front (right)

The function has two peaks. The corresponding true (f^w, δ) Pareto front can be derived as follows. The solution with the best worst-case fitness is clearly the solution at the highest peak ($x = 2.5$) with tolerance level $\delta = 0$. As the tolerance level is increased up to $\delta = 0.5$, the solution $x = 3.5$ remains the optimal solution, but the worst case obviously degrades down to $8/3$. If greater tolerance levels are desired, it is better to switch to solution $x = 5.5$. Although the peak is lower, it is also not as steep, which yields a better worst-case performance for larger tolerance regions. At $\delta = 1.5$, there is no solution that would not have $x = 4$ within its tolerance region $[x - \delta, x + \delta]$, and thus the worst case fitness drops to $f(4) = 1$. If this worst case is accepted, δ can be increased to 3 by choosing $x = 4$, which is the maximally robust solution. So, the Pareto set consists of all solutions $(x = 2.5, \delta \in [0, 0.5])$, $(x = 5.5, \delta \in (0.5, 1.5))$, $(x = 4, \delta = 3)$. The two objectives we would like to optimise are the worst-case fitness and robustness, for this simple test function, we can derive the relationship between those two objectives and can be defined mathematically as $f^w(\delta) = \max_x(f^w(x, \delta))$. For each decision variable, search for the best worst-case fitness with different tolerance levels. The trade-off between those two objectives is depicted in Figure 2.5. For each robustness, we can calculate the corresponding best worst-case fitness. Therefore, the trade-off between the worst-case fitness and robustness can be represented by $f^w(\delta)$ as follow:

$$f^w(\delta) = \begin{cases} -\frac{5}{3}\delta + 3.5 & \text{if } 0 \leq \delta \leq 0.5 \\ -\frac{2}{3}\delta + 3.0 & \text{if } 0.5 < \delta < 1.5 \\ 1.0 & \text{if } \delta = 3.0 \end{cases}$$

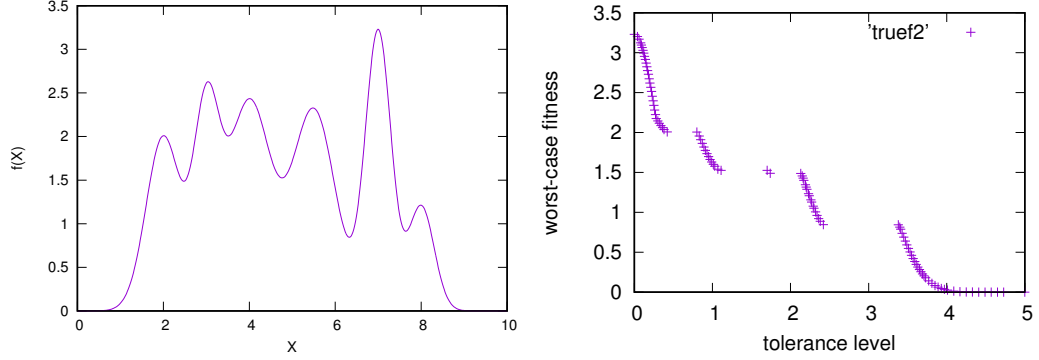


Figure 2.6: The plots of test function 2 (left) and its approximated true Pareto front (right)

2.5.2 Test function 2

The following test function has been taken from Lim et al. (2005), and a plot of this function is shown in Figure 2.6. It has more local optima which makes it more interesting. The true Pareto front shown in Figure 2.6 is approximated by generating a number of equally distributed samples δ , and then for each δ find its optimal worst-case fitness by sampling a number of upper level input designs x_i .

$$f(x) = 2e^{-(x-2)^2/0.32} + 2.2e^{-(x-3)^2/0.18} + 2.4e^{-(x-4)^2/0.5} \\ + 2.3e^{-(x-5.5)^2/0.5} + 3.2e^{-(x-7)^2/0.18} + 1.2e^{-(x-8)^2/0.18}$$

where $0 \leq x \leq 10$.

2.5.3 Test function 3

This is simply a 2-dimensional version of test function 1, and defined as $f(X) = \sum f(x_i)$. It is visualised in Figure 2.7. The function has four peaks, which are $f(2.5, 2.5) = 7.0$, $f(2.5, 5.5) = 6.5$, $f(5.5, 2.5) = 6.5$, $f(5.5, 5.5) = 6.0$. Its worst-case fitness and robustness have been described in the problem definition.

Its true Pareto front (f^w, δ) can be derived as follows. The solution with the best worst-case fitness is the solution at the highest peak ($x_1 = 2.5, x_2 = 2.5$) with tolerance level $\delta = 0$. With the tolerance level increased up to $\delta = 0.5$, the solution ($x_1 = 2.5, x_2 = 2.5$) remains the optimal solution, while its worst-case fitness drops to $16/3$. If the tolerance level continues to increase, the optimal solution will be ($x_1 = 5.5, x_2 = 5.5$) that has the lowest peak. The solution with the lowest peak is not as steep as the other three peaks, and it has a better worst-case

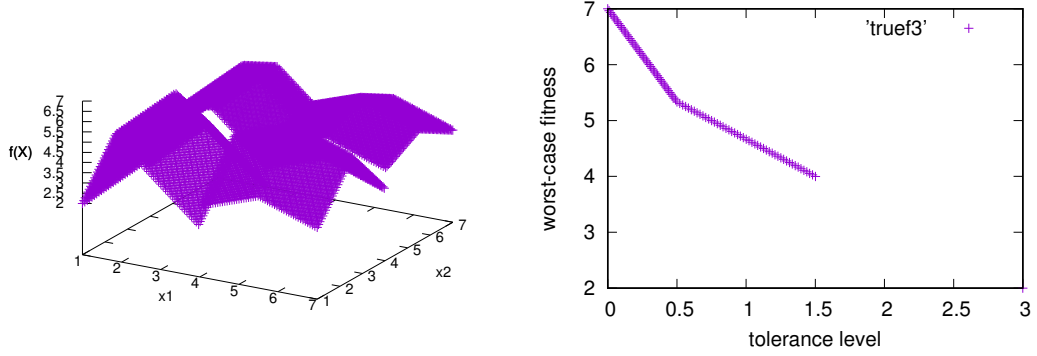


Figure 2.7: The plots of test function 3 (left) and its true Pareto front (right)

fitness for higher tolerance levels. When the tolerance level goes up to $\delta = 1.5$, for each solution can have this tolerance level, its region always include the solution $(x_1 = 4.0, x_2 = 4.0)$, and its worst-case fitness will degrade to $f(4.0, 4.0) = 2.0$. As δ increased to the maximum tolerance level 3.0 with the solution $(x_1 = 4.0, x_2 = 4.0)$, the corresponding worst-case fitness stays at 2.0. Therefore, the true Pareto optimal solution set will be $(x_1 = 2.5, x_2 = 2.5, \delta \in [0, 0.5])$, $(x_1 = 5.5, x_2 = 5.5, \delta \in (0.5, 1.5))$, and $(x_1 = 4.0, x_2 = 4.0, \delta = 3.0)$. So we can get its true Pareto front shown as Figure 2.7.

$$f^w(\delta) = \begin{cases} -\frac{10}{3}\delta + 7.0 & \text{if } 0 \leq \delta \leq 0.5 \\ -\frac{4}{3}\delta + 6.0 & \text{if } 0.5 < \delta < 1.5 \\ 2.0 & \text{if } \delta = 3.0 \end{cases}$$

2.6 Performance measure

The most widely used performance metric for multi-objective optimisation is probably the Hypervolume (Zitzler & Thiele 1998), which measures the volume bounded by a reference point and non-dominated solutions in the objective space. However, in our case, the proposed algorithms may find solutions on either side of the true Pareto front. If the lower level EA does not properly identify the worst case, then the information provided to the upper level EA is too optimistic, individuals will appear better than they are, and solutions to the upper right of the true Pareto front may appear (they do not actually exist, but the algorithm reports them as solution). On the other hand, even if the lower level EA computes the correct worst case, the upper level may not be able to find the best upper level solutions, in which case

the results will appear towards the lower left of the true Pareto front. Hypervolume would count the former error (solutions perceived as too good) as benefit, and is thus not really suitable.

We therefore propose to use the inverted generational distance (IGD) metric (Knowles & Corne 2002). This metric requires a number of targets along the true Pareto front, and then sums up the distances between each target and the closest point in the Pareto front approximation identified by the algorithm. We have chosen 100 equidistant points along the true Pareto front as targets. Because IGD tends to be smaller if the number of solutions in the Pareto front approximation is larger, and our two algorithm variants have different population sizes, we used crowding distance pruning to reduce the number of solutions returned by each algorithm to 100 before computing the IGD metric. For each point, its nearest neighbours in the same front define a cuboid and a crowding distance metric is defined as the average side length of this cuboid. This metric is used in the selection to keep the population diverse, because the larger this metric value, the fewer points in the neighbourhood of this point. Crowding distance pruning means to use the crowding distance as the selection criteria to remove solutions in the population gradually.

2.7 A review of related work

A number of papers (Beyer & Sendhoff 2007, Jin & Branke 2005, Roy 2010, Gaspar-Cunha & Covas 2006, 2008, Bertsimas et al. 2011, 2010) have suggested to use evolutionary algorithms (EAs) to search for robust solutions in optimisation problems. Uncertainties in optimisation problems have been addressed in many application areas (Ide et al. 2015) (Gu et al. 2013) (Thompson 1998) (Greiner 1996) (Wiesmann et al. 1998) (Anthony & Keane 2003a) (Anthony & Keane 2003b) (Chen et al. 2012) (Sebald & Schlenzig 1994). Jin & Branke (2005) provide a survey on evolutionary optimisation with uncertainty. Uncertainty in evolutionary computation is divided into four categories: 1) The fitness function is subject to noise that is often assumed to follow a normal distribution with mean zero and variance σ^2 . The expected fitness function is often approximated by an averaged sum of a number of random samples. 2) The decision variables or environmental parameters are perturbed. Solutions that still work well with slight variations in decision variables are defined as robust solutions. Those solutions are desired rather than global optimal solutions. 3) If the fitness function is approximated it suffers from approximation errors. 4) The fitness function changes over time. Beyer & Sendhoff (2007) makes a survey of robust optimisation. They discuss how to address different kinds of

uncertainties and how to evaluate robust solutions. Ide & Schöbel (2016) introduce a variety of robustness concepts for multi-objective optimisation problems with uncertainty. Goh & Tan (2007b) and Beyer (2000) examine evolutionary optimisation with noisy environments.

A considerable amount of literature has been published on optimisation problems with uncertainty. The previous studies will be discussed in the following sections. The first section reviews single-objective robust optimisation. The second section focuses on multi-objective robust optimisation. The third section will discuss optimisation problems with uncertainty in the constraints. Finally, it introduces active robustness where a parameter can be adjusted to mitigate the disturbance.

2.7.1 Single-objective robust optimisation

Most of the papers to date search for solutions with a good *expected* performance given a distribution of possible disturbances of the decision variables. The main challenge then is to estimate the expected performance efficiently. One of the earliest approaches was probably (Tsutsui & Ghosh 1997, Tsutsui et al. 1996, Tsutsui 1999, Tsutsui & Ghosh 2003) who suggested to simply disturb the decision variables randomly before evaluation. They show that for the limit case of infinite population sizes, this would lead the EA to search the landscape of expected fitness. To get more accurate estimates of the expected fitness also with small population sizes, researchers have proposed explicit averaging over multiple samples (e.g., (Branke 1998, Wiesmann et al. 1998, Branke 2001)) or also to use surrogate functions to avoid costly fitness function evaluations (Paenke et al. 2006). Branke & Schmidt (2005) introduce two fitness estimation methods that are interpolation and regression. Forouraghi (2000) optimises a “signal-to-noise ratio” rather than the expected fitness, which is basically a mean of squared fitness values, penalising variance. Interestingly, Forouraghi (2000) does not pre-specify the distribution of disturbances, but assumes equal distribution in an area that can also be set by the EA. So, the EA specifies a range for each decision variable, rather than a single value.

Beyer & Sendhoff (2006) address optimisation with actuator noise where the noise is added to the design variables. The expected value robustness measure is optimised, and Evolution Strategies (ESs) are suggested to solve the problem where the mutations in ESs play the role of robustness tester.

2.7.2 Multi-objective robust optimisation

While most of the work on EAs for searching robust solutions assumes a single objective function, there are few papers that transfer these concepts to the multi-objective case (Deb & Gupta 2005, 2006, Forouraghi 2000, Saha et al. 2011). Some authors have also used multi-objective evolutionary algorithms (MOEAs) to examine the possible trade-off between solution quality and robustness, again with various definitions of robustness. In addition to the nominal (undisturbed) performance of a solution, Jin & Sendhoff (2003) consider a variance measure, Luo & Zheng (2008) consider an estimate of the gradient in the neighbourhood, and Goh & Tan (2007a) consider the maximum percentage degradation in fitness within a given neighbourhood. They present a robust multi-objective evolutionary algorithm (RMOEA) that aims to evolve the trade-off between Pareto optimality and robustness. They consider the worst-case scenario for each candidate solution and use local search process to find its worst performance. The way they measure the robustness is $f'_i(x) = \frac{\max(f_i(x') - f_i(x))}{f_i(x)}$ that describes the variation degree arising from the worst objective value. In our problem, the robustness is defined as the maximum allowed deviations from the decision variables. We get the actual worst-case fitness with no variation degree constraint.

In (Li et al. 2005, Lim et al. 2005, 2007), robustness is considered as the maximum deviation from the specified decision variables that guarantees that the drop in performance from nominal to realised fitness is no more than some pre-specified threshold. Deb et al. (2009) look at the trade-off between performance and reliability, which is the probability of the solution being feasible.

Deb & Gupta (2005) suggest methods that search for robust solutions in multi-objective optimisation problems. The aim is to find solutions that are less sensitive to small changes in variables. Two different robust multi-objective optimisation procedures are presented to find the robust optimal front rather than the global Pareto-optimal front. They optimise the mean effective objective values that are computed by averaging the objective function values of solutions, instead of the original objective functions. They define two types of multi-objective robust solutions. In type I, the mean effective objective values are defined as $f_j^{eff}(x) = \frac{1}{|B_\delta|} \int_{y \in x+B_\delta} f_j(y) dy$, where $|B_\delta|$ is the volume of the neighbourhood. In type II, they optimise the original objective function by setting the constraints $\frac{\|f^p(x) - f(x)\|}{\|f(x)\|} \leq \eta$. $f^p(x)$ can be the mean effective function value or the worst function value in the vicinity. This means the variations of objective values should be no more than a certain percentage of the original objective function value of a

solution with respect to the pre-defined η . The single-objective robust optimisation problems (SROPs) are extended to multi-objective robust optimisation problems (MROPs) by evaluating the effective objective functions, and the technique is denoted as Eff-MOEA.

Lee & Park (2001) take into account uncertainty in form of variations from the decision variables in engineering optimisation problems. Assume the disturbed decision variable is normally distributed with the mean μ_{x_i} and standard deviation σ_{x_i} . The tolerance band of this decision variable is defined as $a * \sigma_{x_i}$. If $a = 3$, 99.73% of the decision variable exists within $[\mu_{x_i} - 3\sigma_{x_i}, \mu_{x_i} + 3\sigma_{x_i}]$. This is similar to the tolerance level in our problem, but our decision variables are bounded to an allowable range.

Jin & Sendhoff (2003) consider the robustness as an additional objective and the single objective optimisation problem becomes a multi-objective optimisation problem. A trade-off between the performance and robustness is considered. Gunawan & Azarm (2005) introduced sensitivity region concept to measure the multi-objective sensitivity of a design. Considering the objective function contains design variables and parameters, if the variations in objective value is small when the parameter changes, then the design variable is not sensitive to parameter variations. It does not require the parameter distribution so this method also applies to objective functions that are non-differentiable or discontinuous. Li et al. (2005) describe the robust optimal solutions are those solutions that are less sensitive to parameter variations, if the multi-objective optimisation problems involve parameters that are uncontrollable. A new Robust Multi-Objective Genetic Algorithm (RMOGA) is presented in (Forouraghi 2000) to get the trade-off between performance and a robustness index that is defined based on worst-case sensitive region. Luo & Zheng (2008) propose a new method to search for robust solutions by converting a multi-objective robust optimisation problem into a bi-objective optimisation problem, one objective represents the solutions' quality and the other objective is to optimise the solutions's robustness.

For multi-objective optimisation problems in the presence of uncertainty, a conservative method to tackle them is to search for a solution that is robust with respect to best worst-case performance in all possible scenarios. Vasile (2014) search for the optimal design in worst-case scenario as robust solutions. There are also some papers solving minmax problems using coevolutionary algorithms (Jensen 2004) (Jensen 2001) (Branke & Rosenbusch 2008). Evidence-based robust optimisation is introduced in (Alicino & Vasile 2014) to solve multi-objective minmax problems. In (Marzat et al. 2013), Kriging and relaxation is combined to deal with worst-

case global optimisation of black-box functions. Herrmann (1999) aims to solve minmax optimisation problems and find robust solutions that have best worst-case performance with respect to different scenarios.

Inverse multi-objective robust evolutionary (IMORE) design optimisation with uncertainty is proposed by Lim et al. (2005). They consider the worst-case performance for a given performance degradation level d_t . There are two criteria to be optimised, one is nominal fitness and the other is robustness that is defined as the maximum variations for decision variables that allow the performance degradation no larger than the permitted degradation tolerance d_t . Based on those two criteria they search for the trade-off between the nominal fitness and robustness. IMORE is a three-level algorithm, for any decision variable in the first level, the third level returns the worst-case performance and the second level returns the robustness.

Avigad et al. (2005) treats the robustness as the result of delayed decisions for the conceptual design, and a robust non-dominance sorting procedure is involved. Considering uncertainty in multi-objective optimisation problems, the performances of solutions depend on different scenarios. Later, Avigad & Branke (2008) extend the worst-case evolutionary multi-objective optimisation where the number of scenarios is larger, such that for each solution the worst cases will be a set. Branke et al. (2013) introduces Pareto dominance concept to worst-case optimisation problems, where the authors extended the dominance relation to each solution is presented by a set of fitness vector. Two approaches, the expected marginal utility and an indicator based on (Zitzler et al. 2003), are proposed to rank individuals within a front.

(Kuroiwa & Lee 2012) use the worst-case approaches and define three robust solutions for the multi-objective optimisation problem. For each objective function, its worst-case over all scenarios is considered. The uncertain multi-objective optimisation problem will become deterministic by optimising the worst-case of each objective, and the efficient solutions are defined as robust. Barrico & Antunes (2006) applied a degree of robustness concept that is similar to (Lim et al. 2005) to search for robust solutions. For a pre-defined threshold of objective variations, the degree of robustness of a solution is defined as the maximum radius inside a hyper box around it. (Meneghini et al. 2016) propose a coevolutionary algorithm to solve robust multi-objective optimisation problems. They use two populations that represent the solutions and uncertainties, respectively. These two populations compete in the environment.

2.7.3 Reliability

For an optimisation problem with uncertainty in both objective function and constraints, an optimal solution may become infeasible due to the uncertainty. A solution with a small probability of becoming an infeasible is considered as more reliable. The difference between reliability and robustness is that reliability focuses on solving the constraints to make the solution feasible. There are also some papers that consider the uncertainty in constraints. Deb et al. (2009) consider the reliability of solutions. A solution is considered as reliable if it is robust in terms of feasibility with the uncertainty in the decision variables. Ray (2002) pointed out that it is of no practical use to maximise the performance only, because a solution may be too sensitive to small variations. The author proposed to use EAs to maximise three objectives that include a solution's performance, its mean and standard deviation in the neighbourhood. Gupta & Deb (2005) investigate constraints handling in robust multi-objective optimisation.

Deb & Gupta (2006) investigate the two notions in more detail, and they extend them to constrained optimisation problems. They introduce the index robust constraint violation of each solution: $RCV(x) = \sum_{y \in B_\delta(x)} CV(y)$, and the constraint violation of y is defined as $CV(y) = \sum_j \langle g_j(y) \rangle$. The bracket operator $\langle \gamma \rangle$ is defined as $\sum_j \min\{g_j(y), 0\}$. This means that any point in the neighbourhood of a solution violates any constraint, this solution will be considered as infeasible. An index of robust constraint violation of each solution is defined as the the sum of the constraint violations of solutions in the neighbourhood of the candidate solution.

In (Goh & Tan 2007a), the μ GA optimises a multi-objective problem that maximises the worst case objective and the worst constraint violation, and the second criterion also considers the feasibility. Moreover, the memory-based feature of tabu search (TS) is implemented to improve computational efficiency, while the constraints under uncertainty and periodic re-evaluation of archived solutions are used to reduce uncertainty of evolved solutions. Lee & Park (2001) also consider the robustness of constraint functions that is the feasibility of the constraints. A penalty factor is included to control the robustness of the constraint function. On the one hand, the increase of the penalty factor will decrease the possibility that a solution enters into the infeasible region. On the other hand, the objective function value will become worse with the increase of the penalty factor.

2.7.4 Active robustness

The robust optimisation tries to find robust solutions where the decision variable is fixed and the robustness is inherent within the solutions. This kind of robustness is passive. The active robustness is introduced in (Salomon et al. 2014). They consider problems that contain uncertain environmental parameters and adjustable variables that can be modified after the environmental parameters have been revealed (Salomon et al. 2013). For each candidate solution, its performance changes according to the scenario of environment and adjustable variable. Therefore when the environment changes, solutions' performance and robustness can be improved by adaptation. There are two objectives considered, one is the performance and the other is the cost of adaptation. They combine robust and dynamic optimisation to form active robust optimisation. However, they do not consider the cost of adaptation. In (Salomon et al. 2015), the active robust optimisation problem is extended to multi-objective optimisation problems. They consider the active robust optimisation problem as a bi-level optimisation problem. The difference is that the lower level searches for the optimal configurations of the adaptive variables.

Chapter 3

Finding the trade-off between worst-case quality and robustness

This chapter is based on the completed paper about finding the trade-off between worst-case quality and robustness (Branke & Lu 2015). More specifically, we suggest two multi-objective evolutionary algorithms to compute the available trade offs between allowed tolerance level and worst-case quality of the solutions. Both algorithms are 2-level nested algorithms. While the first algorithm is point-based in the sense that the lower level computes a point of worst case for each upper level solution. The second algorithm is envelope-based, in the sense that the lower level computes a whole trade-off curve between worst-case fitness and tolerance level for each upper level solution.

3.1 Motivation of the developed algorithms

A typical problem in engineering is that manufacturing is not able to produce exactly to specification, but instead will introduce some deviations from the design variables. An engineer has to take this into account by allowing for manufacturing tolerances. To our knowledge, no one so far has studied the trade-off between robustness in the sense of acceptable deviation from specified decision variables (tolerance level), and worst-case quality, although this seems of great practical value. In manufacturing, keeping a small tolerance level for a solution is usually expensive. For a certain tolerance level, the engineer would like to know what the acceptable worst-case quality. Therefore, it is of importance to get the trade-off between the

worst-case performance and the tolerance level. For an pre-defined tolerance level, it allows to identify the solution with best worst-case performance from the trade-off. A possible reason may be that determining the worst case is itself a difficult optimisation problem. Searching for the worst-case fitness requires an optimisation algorithm which increase the computational complexity. Some search for the worst-case fitness for a single tolerance level, or get the trade-off between the nominal fitness and robustness by pre-setting the allowed worst-case nominal fitness degradation. We are the first to look at the trade-off between worst-case fitness and robustness, and obtain the optimal solutions in a single run.

In this chapter, we tackle the tolerance/worst-case quality problem by suggesting and comparing two nested MOEAs. We use an evolutionary multi-objective (EMO) algorithm to determine the trade-off between tolerance level δ and worst case performance $f^w(x) = \min_{x' \in [x-\delta, x+\delta]} f(x')$. This problem has first been addressed in (Branke & Lu 2015), where an envelope-based (where the lower level is multi-objective) and a point-based algorithm (lower level is single-objective) were proposed.

The most similar previous approach is the inverse multi-objective robust evolutionary (IMORE) design optimisation as proposed in (Lim et al. 2005) and further refined in (Lim et al. 2007). Its structure is shown as Figure 3.1. IMORE computes the trade-off between nominal fitness (without disturbance) and robustness (here the tolerance level for decision variable disturbance such that the degradation in fitness is no more than a pre-set threshold). This is different from our problem. They maximise the nominal fitness and robustness while we maximise the worst-case performance and robustness. The worst-case performance in IMORE is restricted to a certain level, because the degradation from the nominal fitness to the worst-case performance is bounded to a predefined level. It is a three-level optimisation approach: For each solution, the robustness is evaluated by solving a sequence of worst-case searches for different tolerance levels. Obviously, such a three-level search is computationally very expensive. In Figure 3.2, solution A and B are non-dominated with each other based on IMORE. With the same degradation level, solution A has better nominal fitness but smaller robustness, solution B has a lower nominal fitness but larger robustness. Therefore, solution A and B are non-dominated with each other. Actually, solution A is preferable for any tolerance level, because for the same robustness solution A always has better performance.

We look at a slightly different problem. Rather than searching for the trade-off between nominal performance and tolerance level given a constraint on degradation, we search for the trade-off between worst-case performance and tolerance level.

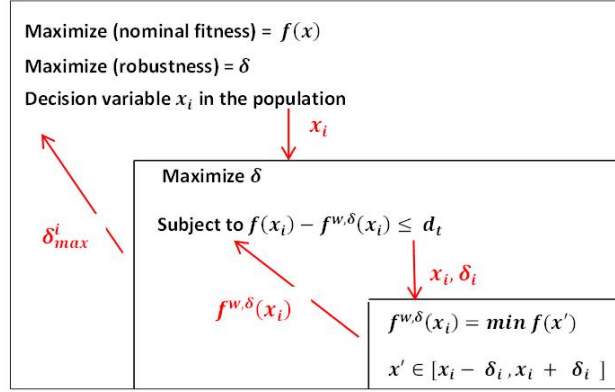


Figure 3.1: IMORE framework

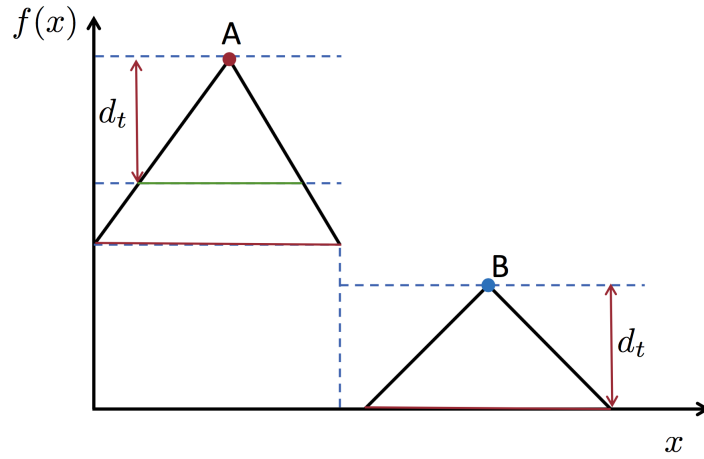


Figure 3.2: Solutions compare based on IMORE

This seems more intuitive and avoids the need to specify an allowable degradation before the optimisation takes place. (Branke et al. 2009) propose portfolio optimisation with an envelope-based multi-objective evolutionary algorithm. In their research, the search space is separated to a set of convex subsets by applying MOEA, and for each subset we solve the problem and obtain an efficient frontier. Afterwards, we merge the partial solutions to form the solutions of the original problem.

We propose two different MOEAs for this problem, one of them is point-based, the other one is envelope-based. While they are both nested, they only work on two levels rather than three as (Lim et al. 2007), which should make them more efficient. The two algorithms are compared empirically on the benchmark problems described in Chapter 2. We also discuss the difficulty of performance metrics for this problem, and conclude that inverse generational distance (Knowles & Corne 2002) is a suitable metric. The chapter is structured as follows. Section 3.2 describes the two algorithms proposed to tackle the problem. The empirical evaluation can be found in Section 3.3. The chapter concludes with a summary about the proposed point-based and envelope-based algorithms.

3.2 Proposed solution approach

We propose two alternative approaches to tackle this problem. Both algorithms are two level nested MOEAs. Worst-case fitness and robustness are the objectives to be maximised in the upper level. The first algorithm is point-based in the sense that for each upper level solution the lower level returns a point of the worst case fitness. The second algorithm is envelope-based in the sense that for each upper level solution, the lower level obtains a whole trade-off front between worst-case fitness and tolerance level (and we call this front the “envelope”). In the following sections, we provide details of the two proposed algorithms.

We describe the non-dominated sorting that is used in the point-based upper level optimisation algorithm, as well as both upper and lower level optimisation algorithms of the envelope-based algorithm. The non-dominated sorting is applied to the population, and it assigns a rank to each solution in the population based on Pareto dominance. For all the solutions in a population, find all the non-dominated solutions and they are the first non-dominated front. Each solution in this first non-dominated front is given a rank 0. In order to find the next non-dominated front, all the solutions in the first non-dominated front are discounted. Each solution in the second non-dominated front is given a rank 2. Repeat the procedure until all the solutions in the population are given a rank. The solution with lower rank is

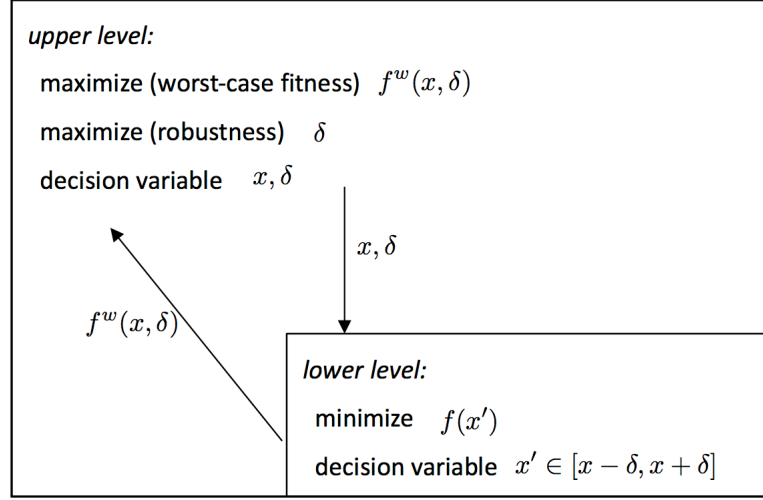


Figure 3.3: Point-based algorithm framework

preferred in the selection.

3.2.1 The point-based nested MOEA

Algorithm 1 Pseudocode for upper level MOEA

```

1: procedure POINT-BASED MOEA
2:   Initialize parent population  $P(x, \delta)$ 
3:   Call lowerEA to evaluate each individual in  $P$ 
4:   for  $j=1$  to  $g$  do  $\triangleright g$  is number of generations
5:     Non-dominate sort  $P$ 
6:     Generate offspring population  $O$  by evolutionary operators
7:     Call lowerEA to evaluate each individual in  $O$ 
8:     Get the union population  $U = P \cup O$ 
9:     Non-dominate sort  $U$ 
10:    Select individuals to form the next generation parent population  $P$ 
11:    Call lowerEA to re-evaluate each individual in the new next generation
    population  $P$ 
12:  end for
13: end procedure

```

The idea of the point-based nested MOEA is relatively straightforward. The general structure is described in Figure 3.3, and the pseudocode for the upper and lower level are provided as Algorithm 1 and Algorithm 2, respectively. The upper level simply optimises two objectives the worst-case quality and robustness as defined above, with x and δ as decision variables. We use an NSGA-II (Deb et al. 2002) type

Algorithm 2 Pseudocode for lower level EA for point-based algorithm

```
1: procedure LOWEREA( $x, \delta$ )
2:   Initialise parent population  $P'$  such that each individual is within  $[x - \delta, x + \delta]$   $\triangleright (x, \delta)$  is upper level individual
3:   Compute the fitness  $f(x')$  of each individual in  $P'$ 
4:   for  $j=1$  to  $g'$  do  $\triangleright g'$  is number of generations
5:     Generate offspring population  $O'$  by evolutionary operators
6:     Compute the fitness of each individual in  $O'$ 
7:     Get the union population  $U' = P' \cup O'$ 
8:     Sort  $U'$  according to fitness
9:     Select best individuals from  $U'$  to form the next generation parent population  $P'$ 
10:  end for
11:  return best individual in  $P'$   $\triangleright$  lowest  $f$ 
12: end procedure
```

MOEA for this purpose. For evaluating the worst case quality $f^w(x, \delta)$, the lower level EA is called. The lower level is a single objective EA that tries to identify the worst case fitness within the tolerance region $[x - \delta, x + \delta]$ around the individual x . The lower level decision variable is defined as $x' = x + \theta$, where $\theta \in [-\delta, \delta]$.

Note that there is no guarantee that the lower level EA actually finds the true worst case for the given x and δ . If it doesn't, the individual looks more promising to the upper level than it actually is, and thus has a higher probability of surviving in the upper level from one generation to the next. Because NSGA-II is an elitist algorithm that always keeps the best solution, this could lead to a situation where the population fills up with solutions for which the true worst case has not been found, which is clearly undesirable. To prevent this from happening, we re-evaluate the population after every generation (Step 11 in Algorithm 1). If the worst case found in the re-evaluation is worse than the one found previously, it is adopted, otherwise it is discarded. The re-evaluation process makes sure that individuals surviving over several iterations are tested again and again, ensuring that their worst-case fitness values are very accurate.

3.2.2 The envelope-based nested MOEA

The general structure of the envelope-based nested MOEA is shown in Figure 3.4, with the pseudocode for the upper and lower level described as Algorithm 3 and Algorithm 4, respectively. In the upper level, the envelope-based MOEA represents an individual only by x (not x and δ as the point-based MOEA introduced above). But in the objective space, each individual is actually represented by a partial

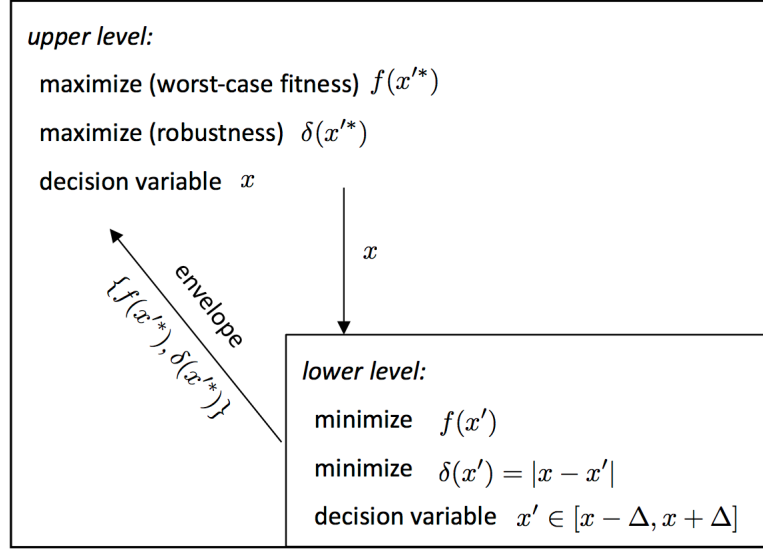


Figure 3.4: Envelope-based algorithm framework

Algorithm 3 Pseudocode for upper level MOEA of envelope-based algorithm

```

1: procedure ENVELOPE-BASED MOEA
2:   Initialise parent population  $P(x)$ 
3:   Call lowerMOEA to evaluate each individual  $P$ 
4:   for  $j=1$  to  $g$  do ▷  $g$  is number of generations
5:     Non-dominate sort  $P$ , using marginal hypervolume as secondary selection
       criteria
6:     Generate offspring population  $O$  by evolutionary operators
7:     Call lowerMOEA to evaluate each individual in  $O$ 
8:     Get the union population  $U = P \cup O$ 
9:     Non-dominate sort  $U$ 
10:    Select individuals to form the next generation parent population  $P$ 
11:    Call lowerMOEA to re-evaluate each individual in the new next genera-
       tion population  $P$ 
12:   end for
13: end procedure

```

Algorithm 4 Pseudocode for lower level MOEA of envelope-based algorithm

```
1: procedure LOWERMOEA( $x$ )
2:   Initialise parent population  $P'$  ( $x', \delta$ )  $\triangleright x$  is upper level individual,
    $x' \in [x - \Delta, x + \Delta]$ ,  $\Delta = \min(x - x^{\min}, x^{\max} - x)$ 
3:   Compute the objective values of each individual in  $P'$ 
4:   for  $i=1$  to  $g'$  do  $\triangleright g'$  is number of generations
5:     Non-dominate sort  $P'$ 
6:     Generate offspring population  $O'$  by evolutionary operators
7:     Compute the objective values of each individual in  $O'$ 
8:     Get the union population  $U' = P' \cup O'$ 
9:     Non-dominate sort  $U'$ 
10:    Select individuals to form the next generation parent population  $P'$ 
11:  end for
12:  Add to population individual ( $\min\{f(P')\}, \Delta$ )
13: end procedure
```

Pareto front, which we call “envelope”, that has been generated by the lower level MOEA. The idea is somewhat reminiscent to the envelope-based MOEA for portfolio optimisation proposed in (Branke et al. 2009), which used parametric quadratic programming to generate an envelope on the lower level.

In our case, the lower level MOEA generates a Pareto front of worst-case $f^w(x, \delta)$ vs. δ trade-off for a particular solution x from the upper level by running an MOEA with the minimisation of $f(x')$ and $\delta(x') = |x - x'|$ as the objectives, and x' as decision variable. To ensure that the tolerance region only contains feasible individuals, x' is restricted to lie within $[x - \Delta, x + \Delta]$, where $\Delta = \min\{x - x^{\min}, x^{\max} - x\}$ is the minimum distance of x from either end of the feasible space. The lower level decision variable is defined as $x' = x + \theta$, where $\theta \in [-\Delta, \Delta]$. The lower level search space is determined by the upper level solution. Because the lower level aims to find the worst-case performance for an upper level solution, we minimise the fitness value and the robustness. The robustness depends on the lower level variable for a specific upper level solution. For each upper level solution, its lower level envelope can be represented by $(f(x'^*), \delta(x'^*))$. The lower level returns an optimal solution set x'^* , and this x'^* depends on the upper level solution. For different upper level solutions, there are different optimal lower level solution sets. In other words, the envelopes are $f^w(x, \delta)$ for a particular x with different tolerance levels, except the flat line where it has the same worst-case fitness value but greater tolerance levels. Additionally, if we maximise the robustness in the lower level, it will return a single point with the maximum robustness and worst-case performance. What is different from point-based algorithm is that the tolerance level is obtained from the lower

level and decided by the lower level decision variables.

Note that given the minimisation of $f(x')$ and $\delta(x')$ as objectives, an individual with maximal tolerance level Δ would be dominated by the solution representing the worst case solution within $[x - \Delta, x + \Delta]$ (unless the worst case is actually on the boundary of the feasible space). However, a decision maker would prefer this solution to the worst case solution, as it provides a larger tolerance level with the same worst case within the tolerance level. To resolve this issue, at the end of each lower level MOEA run, we add to the population an “artificial” individual with the worst fitness of all the individuals in the population, and the maximum allowed $\delta = \Delta$ (see Step 12 in Algorithm 4).

In the upper level we maximise $f(x'^*)$ and $\delta(x'^*)$. Here the tolerance level is a function of x' for a particular upper level solution x . The envelope represents a set of worst-case fitness for different robustness determined by the lower level. In each generation, all the envelopes are combined to form the overall Pareto front. Selection has to be done based on the envelopes. In our case, we use a similar procedure as the non-dominated sorting, where an individual/envelope is counted in the first rank if it has at least one point non-dominated in the union of all envelopes. Each upper level solution has a Pareto front in the objective space, two solutions are non-dominated if both have contribution to the overall Pareto front. To rank individuals in the same non-domination front (done by crowding distance in NSGA-II), we have decided to equate the fitness of an envelope with its marginal hypervolume, i.e., the amount the hypervolume would reduce if this individual/envelope were removed from the population.

Each upper level solution is represented by an envelope. The non-dominance relationship can be extended to the upper level solutions. We use two figures to explain this concept. Solution 1 and 2 are represented by blue squares and red triangles, respectively. Considering the upper level maximises two objectives f_1 and f_2 , solution 1 is dominated by solution 2 from Figure 3.5. Another example is shown in Figure 3.6, the dashed line indicates the overall Pareto front in the upper level. Both solutions 1 and 2 have contribution to the overall Pareto front, and these two solutions are defined as non-dominated with each other.

In our envelope-based algorithm, for a number of solutions that have the same rank, the marginal hypervolume of an upper level solution is used to evaluate how important it is. For a set of non-dominated solutions, the hypervolume is represented by the area dominated by all these solutions. A solution’s marginal hypervolume is defined as the area dominated by this solution excluding the other non-dominated solutions (Branke et al. 2008). As explained above, for an upper

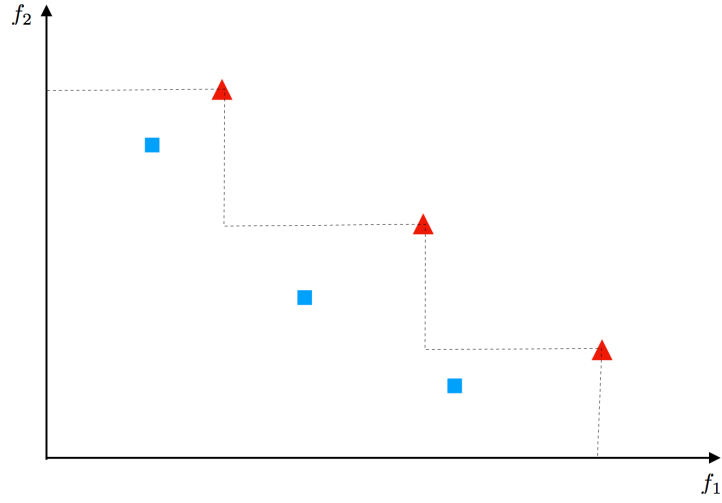


Figure 3.5: The comparison of two upper level solutions, and the solution represented by square is dominated by the solution represented by triangle

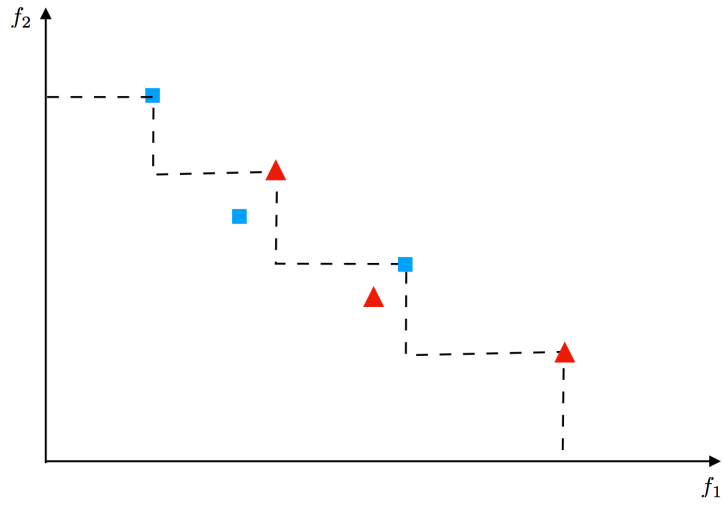


Figure 3.6: Two non-dominated upper level solutions

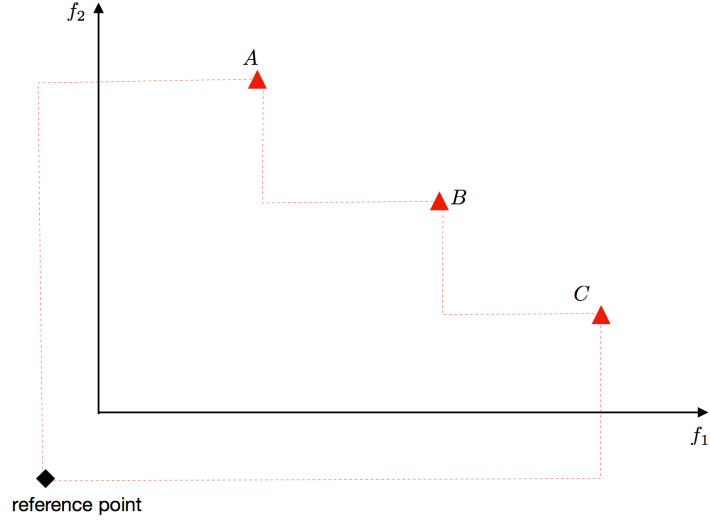


Figure 3.7: Hypervolume of a Pareto front for a two objectives maximisation problem

level solution, its marginal hypervolume is defined as the amount of the hypervolume would reduce if it is removed. For a Pareto front that maximises two objectives, solutions A-C are the non-dominated solutions. The hypervolume is shown as the area bounded by the reference point and non-dominated solutions in Figure 3.7. The reference point is user-defined, in our case, the reference point setting is described as follow. Assume we have a two objectives maximisation problem, find the minimum objective value for each objective. Extend each minimum objective value slightly to its minimisation direction. The intersection point is set as the reference point to calculate hypervolume.

Now we explain the marginal hypervolume using Figure 3.8. Solutions 1 and 2 are non-dominated with each other and have the same rank. The Pareto front is formed by the two solutions. The total hypervolume is shown as the area between the reference point and the six points. If solution 1 is removed, then the hypervolume would reduce the amount of the blue shaded area. This amount is the marginal hypervolume of solution 1. Solutions with greater marginal hypervolume are likely to have more contributions to the overall Pareto front. They are considered to be more important with more chance to be preferred in the upper level selection.

In the envelope-based algorithm framework, given an upper level solution, the lower level algorithm obtains a Pareto front that is defined as an envelope. It is clear that each upper level solution maps to an envelope in the objective space. Non-dominate sort the upper level solutions will be the same as non-dominate sort

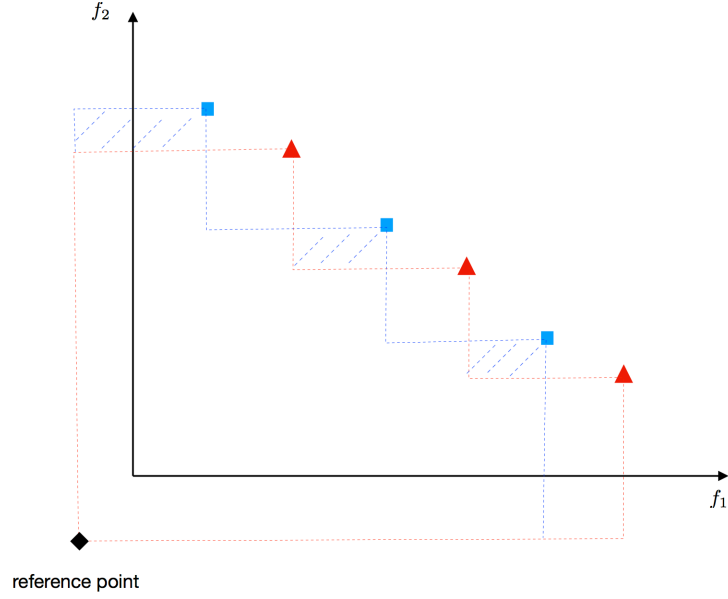


Figure 3.8: Marginal hypervolume of upper level solutions

a set of envelopes. We describe the non-dominated sorting in the upper level population. Firstly, combine all the envelopes and find the overall Pareto front, for those envelopes who have more than one point contributing to the overall Pareto front, they are non-dominated with each other. Those corresponding upper level solutions are the first non-dominated front, each upper level solution in this first non-dominated front is assigned a rank 0. Then find the next non-dominated front, all the upper level solutions in the first non-dominated front will be discounted. Assign a rank 1 to each upper level solution in the second non-dominated front. Repeat this procedure until each individual in the upper level solution is assigned a rank. In order to compare the non-dominated upper level solutions, we introduce marginal hypervolume measure the quality. An upper level solution is considered to better if it has a greater marginal hypervolume.

As in the point-based algorithm, a lower level run that does not identify the true (worst case) trade-off might appear attractive in the upper level. So all surviving individuals are re-evaluated in every generation. The new envelope (obtained by re-evaluation) is combined with the old one (before re-evaluation), and the best $(f(x'), \delta)$ combinations are kept based on a non-dominated sorting and crowding distance calculations.

3.3 Empirical evaluation

In this section, we empirically compare the two proposed algorithms using a simple test function for which we can easily derive the optimum, and a test function taken from the literature. The two algorithms are further compared using a 2-dimensional test function for which its true Pareto front between the worst-case quality and robustness is known.

3.3.1 Parameter settings

We identified suitable parameter settings for each algorithm by some preliminary test runs. The number of solutions in the Pareto front approximation of the point-based algorithm corresponds to the number of individuals in the upper-level MOEA, whereas for the envelope-based MOEA, each individual in the upper level may contribute an entire envelope to the Pareto front approximation. Thus, clearly, the upper level population size of the point-based MOEA needs to be larger than for the envelope-based MOEA. On the other hand, the lower level population size needs to be larger for the envelope-based MOEA, as it needs to generate an entire envelope, rather than only search for a worst case. After some testing, we considered the parameter settings reported in Table 3.1 to be suitable. Table 3.2 displays the parameters setting for 2-dimensional test function 3. Note that these setting lead to the same total number of function evaluations for both algorithms.

Both algorithms on both levels use binary tournament for mating selection, arithmetic crossover for one dimensional problems and SBX crossover for two dimensional problems. Our algorithms are elite based, as generations proceeds more and more good solutions are kept. If the crossover rate is set low, it is unlikely to generate new solutions. Therefore, the crossover is set as in the table in order to keep the diversity of the population. As for mutation, we apply Gaussian mutation with standard deviation of 20% of the search range. The probability of mutating a variable is set as $1/\text{number of variables}$ (Mühlenbein & Schlierkamp-Voosen 1993). For a solution with larger dimensions, the mutation probability is set relatively small, because the mutation in one dimension would produce a new solution. However, for a solution with smaller dimensions, for instance a one-dimensional solution, if the mutation probability is set low, it is more likely that there is no mutation on this solution. Then no new solution would be produced, the algorithm is less efficient.

Table 3.1: Parameter setting for TF1 and TF2

	point-based		envelope-based	
	upper level	lower level	upper level	lower level
popsize	100	40	40	100
max generations	250	40	100	100
crossover prob.	0.9	0.9	0.9	0.9
mutation prob.	0.5	0.9	1.0	1.0

Table 3.2: Parameter setting for 2-dimensional TF3

	point-based		envelope-based	
	upper level	lower level	upper level	lower level
popsize	100	40	40	100
max generations	500	100	100	500
crossover prob.	0.9	0.9	0.9	0.9
mutation prob.	0.4	0.9	0.5	0.5

3.3.2 Test results

Simple test function 1

To evaluate the two algorithms proposed, we ran each algorithm 20 times on the simple test function 1 and measured the IGD. Figures 3.9 and 3.10 show the final Pareto front approximation by the run with the median IGD of the point-based and envelope-based MOEAs, respectively. As can be seen, both algorithms find very good approximations to the Pareto front, clearly identifying all three parts that make up the front. The front of the envelope-based MOEA seems to be somewhat more evenly distributed.

Figure 3.11 shows the average IGD of each algorithm over the number of fitness function evaluations. The right side shows the enlarged figure of the left side. This figure clearly demonstrates the superiority of the envelope-based approach. The envelope-based algorithm shows a smaller IGD (better performance) from the beginning (2.91 vs. 12.4), converges more quickly, and maintains its advantage throughout the run, converging to a better solution.

On this problem, the envelope-based MOEA is computationally more time-consuming, presumably because the non-dominated sorting on the lower level is more complex. However, if function evaluations are expensive, this computational cost will be negligible, and the envelope-based MOEA will show its full advantage.

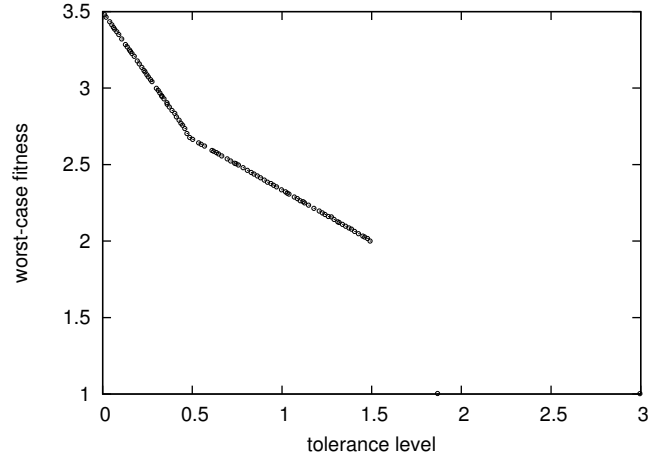


Figure 3.9: The trade-off between worst-case fitness and robustness of TF1 by point-based MOEA

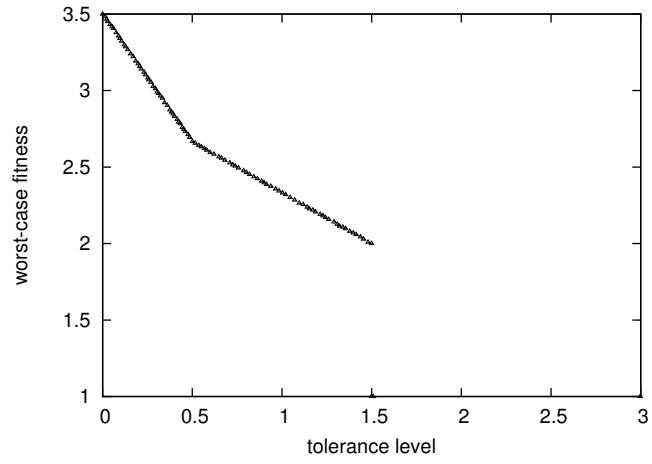


Figure 3.10: The trade-off between worst-case fitness and robustness of TF1 by envelope-based MOEA

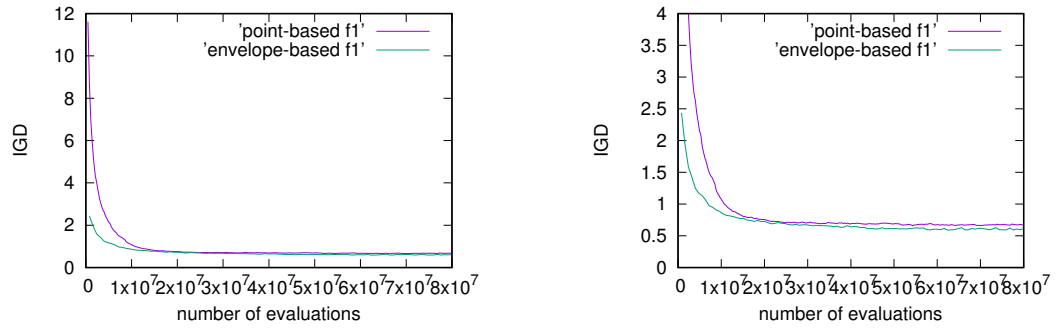


Figure 3.11: The comparison of convergence of TF1

Test function 2 from literature

We used the same parameters as above also on the test function 2 proposed in Lim et al. (2005). Since we don't know the true Pareto front for this problem, we can not compute the IGD. An approximated true Pareto front of test function 2 is described in Chapter 2.

Figures 3.12 and 3.13 show the Pareto front approximation obtained by the point-based MOEA and the envelope-based MOEA, respectively. Conclusions are similar as for the simple test function above. Both algorithms find a very similar front, which indicates that they have converged onto the approximated true optimal Pareto front. Also, the extreme points (minimal tolerance level, $\max\{f^w(x, 0)\} = \max\{f(x)\} = 3.23$ and maximal tolerance level, $\max\{f^w(x, 5)\} = \min\{f(x)\} = 0$) have been identified correctly. Again, the front generated by the envelope-based approach seems to be more uniformly distributed.

Figure 3.14 compares the convergence plots of the point-based and envelope-based algorithms. It can be seen from the figure that the envelope-based algorithm begins from a very small IGD value (about 5.0) and converges very quickly with less than 10^7 evaluations. It maintains this lower IGD value throughout the run. The point-based algorithm starts from a high IGD value (about 25.0) and converges much slower than the envelope-based algorithm. Its IGD value stays higher than the envelope-based algorithm.

2-dimensional test function 3

A 2-dimensional test function 3 is used to evaluate the proposed two algorithms further. This function described in Chapter 2 has four peaks. Its true Pareto front is known, and we can compute the IGD value. Figure 3.15 presents the Pareto front approximation obtained by the point-based MOEA and the envelope-based MOEA, respectively. Compared to the true Pareto front, there are some outliers around tolerance level 1.5. One reason may be that the lower level does not find the true worst case (and search is more difficult in a 2-dimensional space than in the 1-dimensional space). Another reason is that we now have discontinuities where the upper function value is 3, so if the algorithm is slightly off, it is likely to report a value around 3. Both algorithms can find a good Pareto front that converges to the true Pareto front. The Pareto optimal solutions $(x_1 = 2.5, x_2 = 2.5, \delta \in [0, 0.5])$, $(x_1 = 5.5, x_2 = 5.5, \delta \in [0.5, 1.5])$, and $(x_1 = 4.0, x_2 = 4.0, \delta = 3.0)$ have been identified correctly.

Figure 3.16 illustrates the average IGD value of the point-based and envelope-

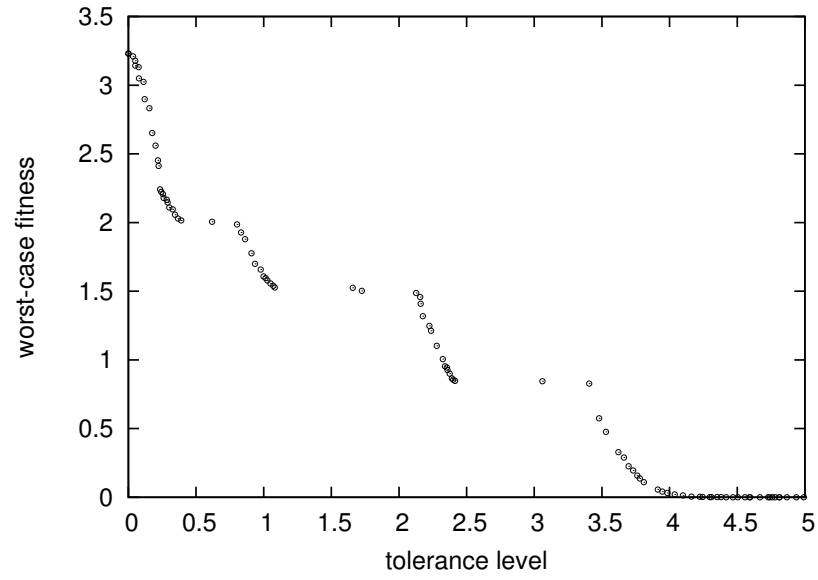


Figure 3.12: The trade-off between worst-case fitness and robustness of TF2 by point-based MOEA

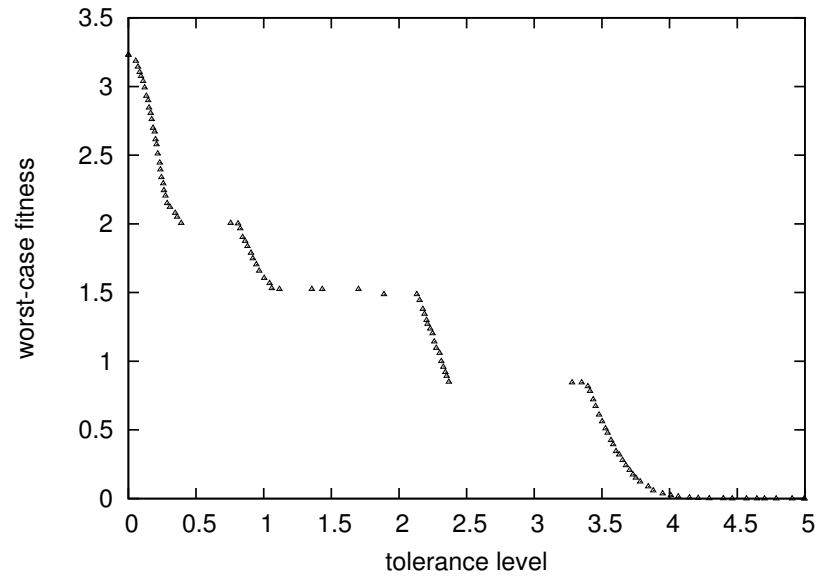


Figure 3.13: The trade-off between worst-case fitness and robustness of TF2 by envelope-based MOEA

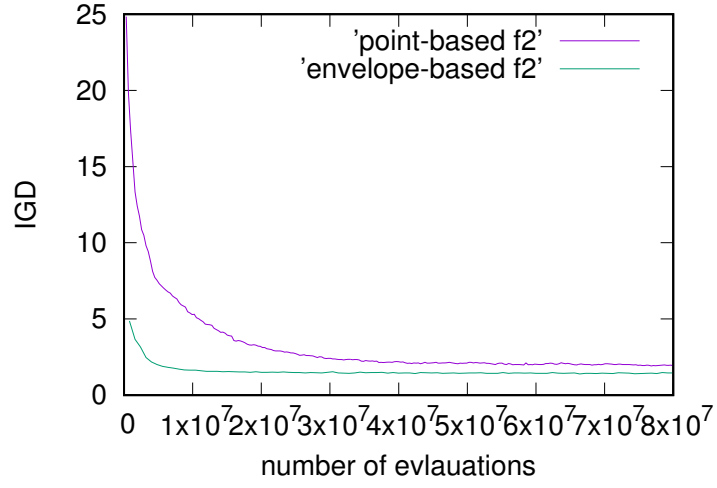


Figure 3.14: The comparison of convergence of TF2

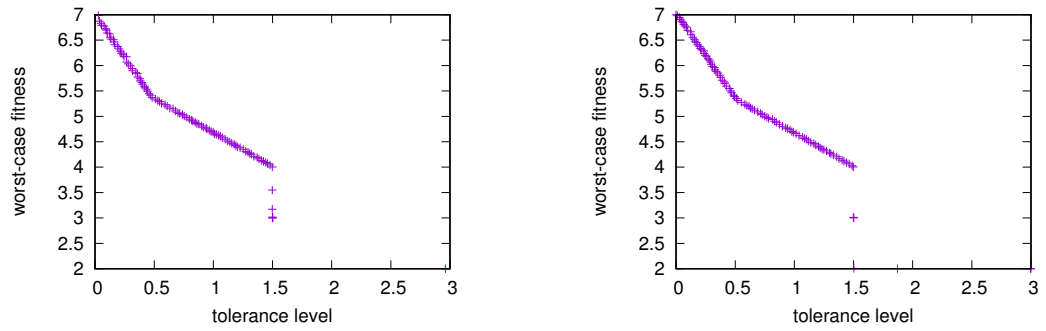


Figure 3.15: The trade-off between worst-case fitness and robustness of TF3 by the point-based MOEA (left) and envelope-based MOEA (right)

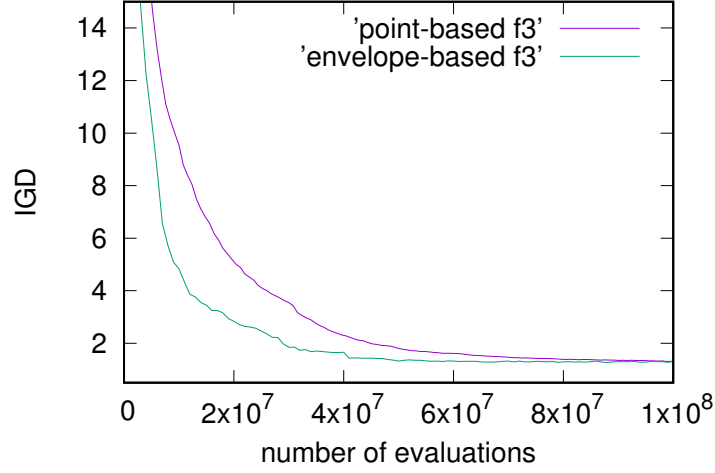


Figure 3.16: The comparison of convergence of TF3

based algorithms over the number of fitness evaluations for test function 3. We have similar conclusions as the above two test functions. Looking at the convergence plot, it is apparent that the envelope-based algorithm has better performance. The envelope-based algorithm starts from a smaller IGD value and converges faster than the point-based algorithm.

3.4 Summary

In this chapter, we investigated the trade-off between worst-case solution quality and tolerance level for a problem with uncertainty in the decision variable space. To solve this problem, we presented two nested multi-objective evolutionary algorithms. One is a point-based MOEA that, in the lower level, simply computes the worst-case performance of an upper level solution. With the envelope-based approach, for each solution, the lower level computes a trade-off envelope between worst-case fitness and tolerance level.

An empirical evaluation of both algorithms demonstrated that at least for the benchmark problems considered, both algorithms can find very good approximations to the true Pareto front. The envelope-based MOEA performed better than the point-based MOEA, with a lower inverse generational distance throughout the run, and a more even distribution of solutions along the front.

Chapter 4

Improving the efficiency of bi-level worst-case optimisation

Our problem defined in Chapter 2 is a special case of a bi-level optimisation problem. In general, bi-level optimisation problems are computationally very expensive. Because a lower level optimiser is called to evaluate each solution in the upper level. This chapter is based on (Lu et al. 2016) but it goes substantially beyond the published paper. We propose and compare several strategies to reduce the number of fitness evaluations without substantially compromising the final solution quality. The chapter is structured as follows. We survey some related work in Section 4.2. Section 4.3 describes six different strategies to reduce the necessary number of function evaluations. The empirical results are discussed in Section 4.4. It concludes with a summary at the end of this chapter.

4.1 The link to bi-level optimisation

In general, a bi-level optimisation problem can be formulated as follow.

$$\min F(x_u, x_l^*) \quad (4.1)$$

$$\text{s. t. } x_l^* \in \operatorname{argmin}\{f(x_u, x_l) : g_i(x_u, x_l) \leq 0, j = 1, \dots, J, x_l \in X_L\} \quad (4.2)$$

$$G_k(x_u, x_l^*) \leq 0, k = 1, \dots, K \quad (4.3)$$

$$x_u \in X_U \quad (4.4)$$

where x_u and x_l are the upper and lower level decision variables, F and f are the upper and lower level fitness function, and G and g are the upper and lower level

constraints, respectively. x_l^* is the optimal lower level decision variables and it could be more than one.

The point-based algorithm can be described as

$$\max \quad f(x'^*), \delta \quad (4.5)$$

$$\text{s. t. } x'^* \in \operatorname{argmin}\{f(x') : x' \in [x - \delta, x + \delta]\}. \quad (4.6)$$

This is a special case of the bi-level optimisation problem with $x_u = \{x, \delta\}$, $x_l = x'$ and $g_1(x, x', \delta) = x' - x + \delta$, $g_2(x, x', \delta) = x - x' - \delta$. In particular, in the upper level we have a multi-objective problem with maximisation of $f(x')$ and δ . The decision variables are x and δ , and for each upper level solution (x, δ) the worst case is identified by solving a lower level single objective optimisation problem. The lower level is a minimisation of $f(x')$ with decision variable x' . The optimal lower level decision variable is represented by x'^* .

Because in bi-level optimisation each upper level solution is evaluated by running a lower level optimiser. The procedure is computationally very expensive. In this chapter, we re-visit the point-based algorithm and suggest several strategies to reduce the necessary number of fitness function evaluations on the lower level.

4.2 Background introduction

Many papers that deal with uncertainty in the decision variables consider expected fitness optimisation rather than worst-case optimisation. A survey on robust design optimisation can be found in (Beyer & Sendhoff 2007). The problem of looking at the trade-off between tolerance level and worst case performance has been introduced in (Branke & Lu 2015) and the baseline algorithm we use here is the point-based algorithm. Since this is formulated as a bi-level problem and computationally expensive, an approach using surrogates is introduced in (Lim et al. 2007). Surrogate-assisted EAs have also been used in (Ong et al. 2006, Zhou & Zhang 2010) for worst-case optimisation. Multi-objective worst-case optimisation is considered in (Branke et al. 2013).

(Kruisselbrink et al. 2010) is based on (Branke 1998) and extend it selecting additional sampling points to enforce the archive points locally well-spread. (Saha et al. 2011) makes improvement of the algorithm proposed by (Deb & Gupta 2006) by reducing the number of function evaluations. Three techniques are proposed to reduce the number of true function evaluations. The first one is to apply bounded archive that in each generation solutions that have been evaluated are stored in

an archive with fixed size. In the second technique after generating the offspring population, only a partial points (a part of the samples) in the neighbourhood are evaluated, and then the healthy offsprings survived from the evaluation are considered more important and re-evaluated within full neighbourhood (all the samples are evaluated). The offsprings and parent population are combined to do the non-dominated sorting and elitist selection. The third technique is to apply smarter sampling. In (Du & Chen 2000), efficient feasibility evaluation methods in robust optimisation are developed to use in engineering design.

There are a number of papers in evolutionary bi-level optimisation (Kalyanmoy Deb 2010) (Deb & Sinha 2009) (Sinha et al. 2014a) (Sinha 2011) (Sinha et al. 2016) (Sinha et al. 2013) (Sinha et al. 2014b). Colson et al. (2007) gives an overview of bi-level optimisation. Oduguwa & Roy (2002) suggested to use genetic algorithm to solve bi-level optimisation problems. In particular (Kalyanmoy Deb 2010) is also suggesting ways to improve the efficiency, including one of the methods we are testing in this paper, namely maintaining the population of a lower level EA so that it can be continued later. However, the special structure of the worst-case optimisation problem allows us to exploit more ways to use upper level information to reduce the fitness evaluations on the lower level. In (Angelo et al. 2014), surrogate models are combined with Differential Evolution in order to solve bi-level problems.

4.3 Algorithm and New Strategies for point-based algorithm

In this section, we will first explain our multi-objective bi-level evolutionary algorithm and then introduce six different strategies for saving fitness function evaluations.

4.3.1 Worst case bi-level evolutionary algorithm

Algorithms 1 and 2 show the pseudocode for the upper and lower level EA, respectively. As can be seen, the upper level is a multi-objective NSGA-II type EA, the lower level is a standard single objective EA. Every individual on the upper level is evaluated by running a lower level EA. Line 11 of Algorithm 1 shows that every individual surviving to the next generation is re-evaluated, and its fitness is updated if a new worst case is found. This is done to ensure that the worst case of solutions surviving over several generations is reliable. Otherwise, a solution for which the lower level was unable to find the true worst case might look deceptively good on

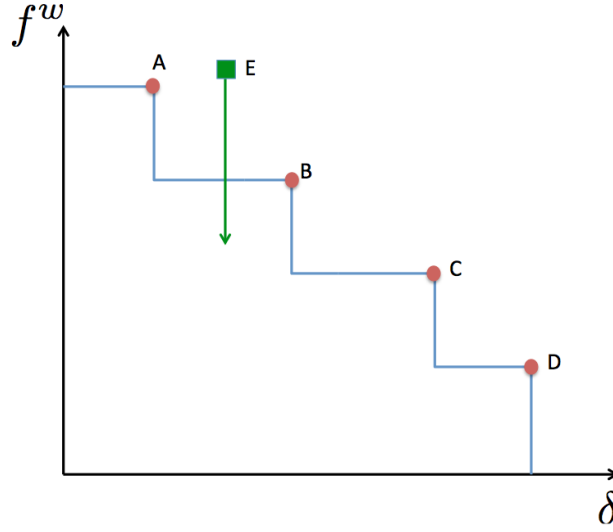


Figure 4.1: Strategy I: If worst case found by lower level enters current dominated region of upper level, the lower level is aborted.

the upper level EA and misguide the search. Obviously, this re-evaluation adds to the computational cost.

4.3.2 Strategies to save fitness evaluations for point-based algorithm

Strategy I : Exploit upper level information for selection and stopping criteria at the lower level

In worst case optimisation, the upper and lower level both use the fitness function f , but for the upper level it is an objective to maximise, whereas the lower level tries to minimise it. This contradiction between the upper and lower levels can be exploited to prematurely stop lower level optimisation runs if it is apparent that the corresponding upper level solution would not be interesting. The concept is visualised in Figure 4.1. Let us assume the lower level is trying to find the worst case of solution E . Then, this solution's worst case objective moves down (lower f^w) during lower level optimisation. If it enters the current dominated region of the upper level population (depicted by the red solutions $A - D$), then we know solution E will be dominated in the upper level, and not contribute significantly to the upper level search, so we can choose to abort the lower level run prematurely. Note that we only do this for a solution's first lower level evaluation, not for re-evaluation, because the purpose of re-evaluation is to make sure we really found the worst case.

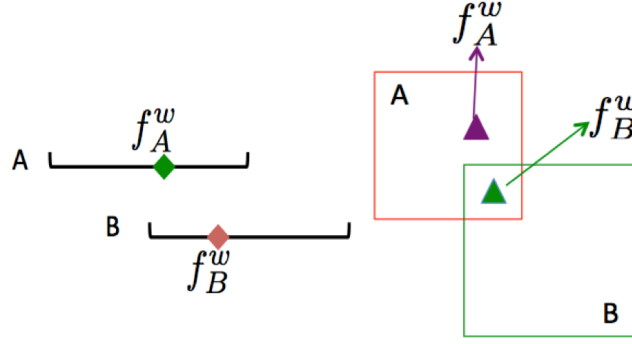


Figure 4.2: Strategy II: Update of worst cases using information from neighbours. Left part shows a one-dimensional example, right part shows a two-dimensional example.

Strategy II: Use neighbours to update worst case fitness

An important but computationally expensive step in the bi-level EA is the re-evaluation of all surviving individuals at the end of each upper level generation. In Strategy II, we propose to replace the re-evaluation by exploiting neighbours to update worst-case information instead. Consider the example depicted in Figure 4.2 which shows two solutions A and B and their corresponding worst cases f_A^w and f_B^w as found by the lower level EA, respectively. The point corresponds to worst case f_B^w lies in the disturbance region of solution A . So, if $f_B^w < f_A^w$, then we know that f_A^w can not be the true worst case of solution A , and we can replace f_A^w by f_B^w as a more realistic estimate of A 's worst case. Because EAs evaluate many solutions in promising regions of the search space, a solution's worst case estimate should quickly become accurate even without re-evaluation.

Strategy III: Skip re-evaluation if it does not improve worst-case estimate

If re-evaluating a solution did not identify a new worst case, we can be more confident that the worst case we found previously is accurate. Strategy III chooses to skip re-evaluation for this solution in the following generation.

Strategy IV: Lower level smart initialisation

The purpose of re-evaluation is to make sure the lower level really correctly identified the worst case. In the baseline algorithm, the lower level is re-started from scratch for re-evaluation. Strategy IV suggests to keep the population at the end of a lower level run in memory, and re-start the search from this population if the corresponding upper level solution is to be re-evaluated. This has previously been proposed in

Kalyanmoy Deb (2010). Note that this may risk getting stuck in a local optimum. Since it does not make much sense to continue running the algorithm once it has converged, we abort the lower level EA in case the improvement over the the past 5 generations was less than 0.001.

Strategy V: Make lower level generations adaptive to δ

For each upper level solution (x, δ) , the lower level searches for its worst-case fitness in the search space defined by this upper level solution. For solutions with smaller tolerance level, the lower level search space is relatively smaller. Therefore, it requires less number of generations in the lower level searching process.

Strategy V thus proposes to reduce the number of generations of the lower level EA for smaller δ . Specifically, we chose to set

$$Generations = \max \left\{ 10, \left\lceil maxGen \times \frac{2\delta}{\delta^{max} - \delta^{min}} \right\rceil \right\}$$

This formulation represents a linear relationship between the generation size and the tolerance level δ , where the minimum tolerance level δ^{min} is 0 and the maximum tolerance level δ^{max} is a constant determined by the decision variables limits. The generation size of the lower level varies with the tolerance level δ .

Strategy VI: Adjust lower level population size to δ

As described above, for an upper level solution with smaller tolerance level, the search space is smaller. Therefore, it requires less number of evaluations in the lower level optimisation algorithm to find the optimal solutions. Instead of reducing the number of generations as in Strategy V, Strategy VI reduces the population size of the lower level EA for small δ . We chose a function that has a minimum population size of 1 for $\delta = 0$, and a maximum population size equal to the usual population size for $\delta = \delta^{max}$, which for the test functions considered later results in

$popSize = \lceil (maxPopSize - 1)/\delta^{max} \times \delta + 1 \rceil$ in case of a single decision variable, and $popSize = \lceil (maxPopSize - 1)/(\delta^{max})^2 \times \delta^2 + 1 \rceil$ for the case of two dimensional problems. For one dimension problems, we assume there is a linear relationship between the maximum population size and the tolerance level, while for the two-dimensional problems we set a quadratic relationship.

Table 4.1: Parameter setting for standard point-based algorithm on TF1 and TF2

	UpperLevel	LowerLevel
popsize	100	40
max generations	250	40
crossover prob.	0.9	0.9
mutation prob.	1.0	0.9

Table 4.2: Parameter setting for standard point-based algorithms of TF3

	UpperLevel	LowerLevel
popsize	100	40
max generations	500	100
crossover prob.	0.9	0.9
mutation prob.	0.5	1.0

4.4 Empirical Results for point-based algorithm

The following test results are based on TF1, TF2 and TF3 mentioned in Chapter 2. Firstly, we will run the algorithm with different strategies with a fixed number of generation size. The total number of evaluations would be reduced compared to the baseline algorithm. Later, the algorithm with each strategy is run with a fixed number of evaluations to show how the methods converge to an approximated Pareto front.

4.4.1 Parameter settings

For the 1-dimensional problems TF1 and TF2, the upper level MOEA was run for 250 generations and used a population size of 100 crossover and mutation probabilities of 1.0 and 0.9, respectively. The lower level EA, uses 40 generations, population size 40, and crossover and mutation probability of 0.9. For the 2D problem TF3, we increased the number of generations to 500 on the upper level and 100 on the lower level. The parameters settings for standard point-based algorithm are displayed in Table 4.1 and Table 4.2.

For performance evaluation we use the Inverse Generational Distance (IGD) because, as explained in (Branke & Lu 2015), it penalises both, if solutions worse than the true Pareto front have been found, and if solutions seemingly better than the true Pareto front are found (which can happen because the lower level is not guaranteed to find the true worst case). All results reported are averages over 20 runs.

Table 4.3: Relative function evaluations and final IGD value when each of the six different strategies are used individually

Strategy	TF1		TF2		TF3	
	Evals	IGD \pm std.err.	Evals	IGD \pm std.err.	Evals	IGD \pm std.err.
Standard	100.0%	0.763 \pm 0.010	100.0%	2.865 \pm 0.079	100.0%	1.265 \pm 0.016
I	54.74%	0.773 \pm 0.012	54.22%	2.913 \pm 0.071	56.43%	1.265 \pm 0.016
II	50.00 %	0.697 \pm 0.012	50.00 %	3.725 \pm 0.054	50.00%	1.420 \pm 0.021
III	74.81 %	0.730 \pm 0.012	74.36%	3.114 \pm 0.078	72.30%	1.431 \pm 0.025
IV	60.87 %	0.775 \pm 0.015	64.57%	2.897 \pm 0.066	52.57%	1.477 \pm 0.021
V	55.67 %	0.790 \pm 0.012	87.13%	2.816 \pm 0.063	49.82 %	1.340 \pm 0.085
VI	31.30 %	0.791 \pm 0.013	47.87%	3.217 \pm 0.059	15.73%	1.387 \pm 0.023

Table 4.4: Combining Strategy I with any of the other five strategies

Strategy	TF1		TF2		TF3	
	Evals	IGD \pm std.err.	Evals	IGD \pm std.err.	Evals	IGD \pm std.err.
Standard	100.0%	0.763 \pm 0.010	100.0%	2.865 \pm 0.079	100.0%	1.265 \pm 0.016
I+II	5.62%	0.735 \pm 0.009	14.26%	3.261 \pm 0.023	6.26%	1.511 \pm 0.020
I+III	28.64%	0.737 \pm 0.009	42.69%	2.822 \pm 0.066	27.91%	1.400 \pm 0.018
I+IV	9.71 %	0.741 \pm 0.011	18.76%	2.506 \pm 0.047	7.43%	1.449 \pm 0.018
I+V	38.50%	1.031 \pm 0.022	63.98%	5.153 \pm 0.159	32.46%	1.283 \pm 0.017
I+VI	20.60%	0.805 \pm 0.016	59.15%	2.736 \pm 0.069	10.84%	1.471 \pm 0.026

4.4.2 Test results and analysis

We will look at the effect of each of the above six methods individually on different test functions, and then try combinations of different methods. In addition to figures showing the reduction of IGD over fitness evaluations, we show tables that report, for each method, the number of fitness evaluations needed (in percentage of the baseline algorithm) and final IGD.

Individual effects

The results of using each of the above six strategies individually is displayed in Table 4.3. They are relatively consistent across test problems. Aborting lower level runs when they result in upper level dominated solutions saves almost 50% of the evaluations. Replacing re-evaluation by neighbourhood update (Strategy II) always saves exactly 50% of the evaluations. Strategy III (skipping some re-evaluations) saves only about 25%. Smart initialisation and early stopping in case of convergence yields between 36% and 48%. The biggest differences between test problems can be found in the techniques that adjust the number of generations or the population size to the size of the lower level search space. In TF3 this seems to yield the greatest

savings, whereas savings in TF2 are relatively modest. Obviously this depends on how many Pareto optimal solutions with large δ exist in the upper level, and for TF3 there are relatively few, resulting in large savings. Reducing the population size seems to reduce the number of fitness evaluations by more than reducing the number of generations, although this certainly depends on the chosen parameter settings.

While the savings in terms of number of function evaluations are massive (up to 85%), some of the methods do not suffer substantially in terms of obtained IGD. On TF1, all methods work quite well, although the last two strategies (adapting the lower level generations or population size to δ) are worst. Reducing the number of generations seems to work slightly better than reducing the population size across all three test functions. Strategy II (replacing re-evaluation by neighbourhood update), despite saving 50% of function evaluations, works better than the baseline algorithm on TF1. The reason is that in TF1, all the Pareto optimal solutions have one of three x values, so there are many solutions with overlapping neighbourhoods which allows Strategy II to work particularly well. And because we can update fitness based on neighbourhood *before* selection, whereas re-evaluation is done only *after* selection, it really can do better. Unfortunately, Strategy V is significantly worse than the baseline algorithm on TF2 which has substantially more peaks.

Combinations of strategies

Several of the strategies can be combined. Table 4.4 reports on results of combining the early abortion of unpromising runs (Strategy I) with each of the other five strategies. The obtained savings in the number of fitness function evaluations is very remarkable and up to 95% on TF1 for the case of combining Strategy I with Strategy II (abortion of lower level runs for unpromising solutions and replacing re-evaluation by neighbourhood update). Note that the savings of Strategy I are independent of re-evaluation, as Strategy I is only applied during an individual's first evaluation, so combining Strategy I (only applied to first evaluation) and Strategy II (to get rid of re-evaluation) is complementary. Looking at the IGD of Strategy I+II, while the IGD is even lower than the standard algorithm for TF1, it is substantially higher for TF2 and TF3. Combination of Strategies I+IV seems to work well on TF3, but poorly on TF1 and TF2 (has a very high standard error on TF2).

Table 4.5: Parameter setting for TF1 and TF2

	UpperLevel	LowerLevel
crossover prob.	0.9	0.9
mutation prob.	0.5	0.9

4.5 Test results for a given number of fitness evaluations

In this section, we adjust the parameters setting displayed in Tables 4.5 and 4.6. Run the algorithms with a given number of fitness evaluations of 8×10^7 for TF1 and TF2. The test results of TF1 is shown in Figure 4.3 and 4.4. In general, the test results shows better performance by adjusting the mutation probability. This could be explained that a good mutation probability is given by $1/\text{number of decision variables}$ (Mühlenbein & Schlierkamp-Voosen 1993).

The test results for TF2 by applying single strategy is described in Figure 4.5. The left side shows the complete convergence plot, in order to see the difference more clearly another convergence plot is displayed on the right side. What should be noticed is that by making the lower level population size adaptive to tolerance levels, the algorithm converges earlier but with a worse IGD value than the standard algorithm. This is due to that TF2 has a larger search range compared with TF1. With a small number of evaluations with a larger search space, the lower level is unlikely to find the true worst-case fitness. At the end of each generation, the algorithm may find more solutions appear better but have worse performance after re-evaluation, and these solutions will be discarded and in the end the algorithm tends to find less optimal solutions. This will make the IGD value greater. One possible way is to adjust the lower level population size and generation size because TF2 has a larger search space and it makes sense to increase the size. The convergence plot of algorithm by combining strategy I with the other is shown in Figure 4.6. Generally, the algorithms have better performance with the combination of two strategies. The combination of strategy I and VI have even worse convergence IGD that is consistent with the convergence plot applying strategy VI individually.

By applying those strategies described above, it is desired to obtain a good Pareto front between the worst-case quality and robustness given a small number of fitness evaluations. The strategies makes our algorithms more attractive when there is a limited computational budget.

Table 4.6: Parameter setting for standard point-based algorithms of TF3

	UpperLevel	LowerLevel
crossover prob.	0.9	0.9
mutation prob.	0.4	0.5

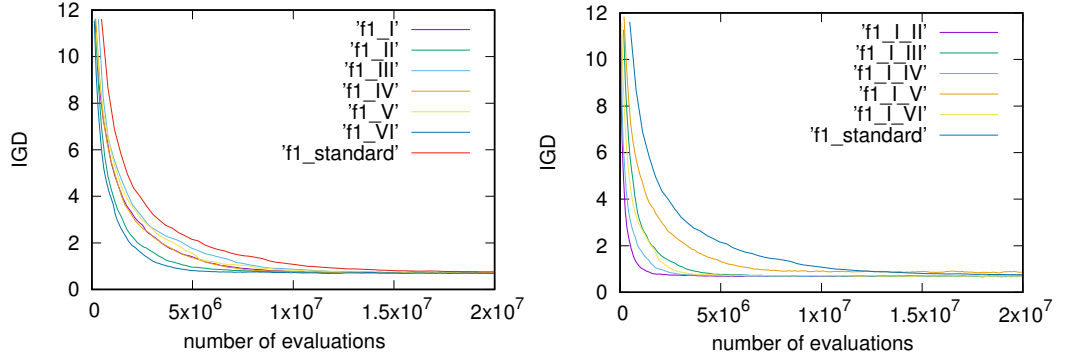


Figure 4.3: Individual effects by applying those strategies on TF1 .

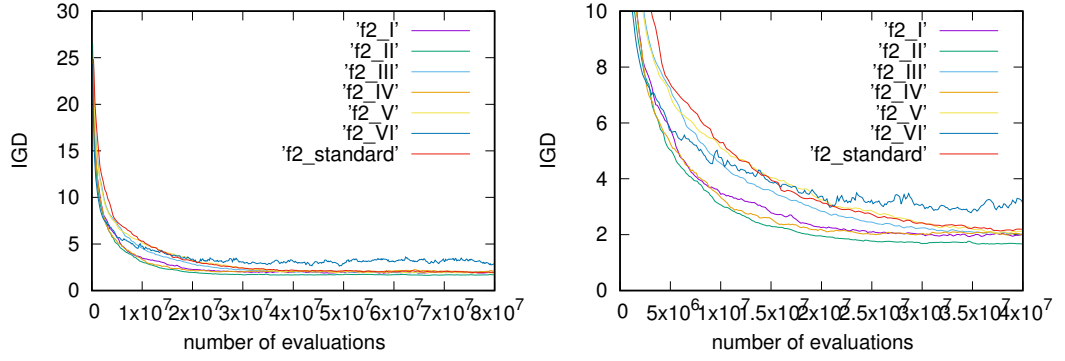


Figure 4.5: Individual effects by applying those strategies on TF2 .

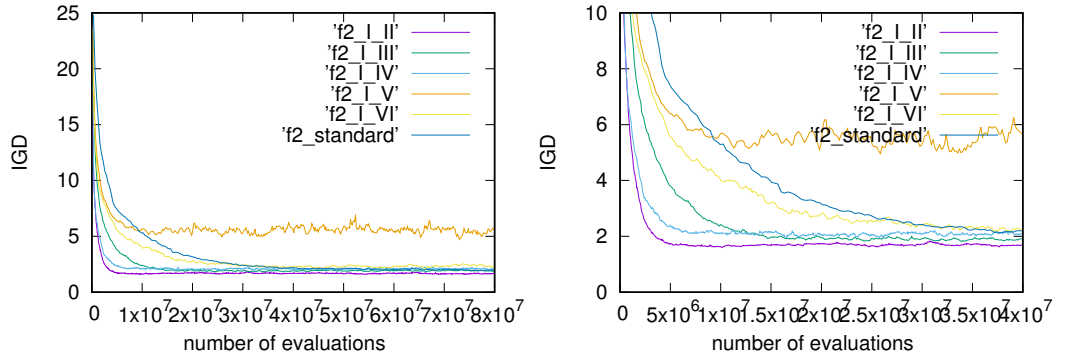


Figure 4.6: Effects of combining Strategy I with other on TF2 .

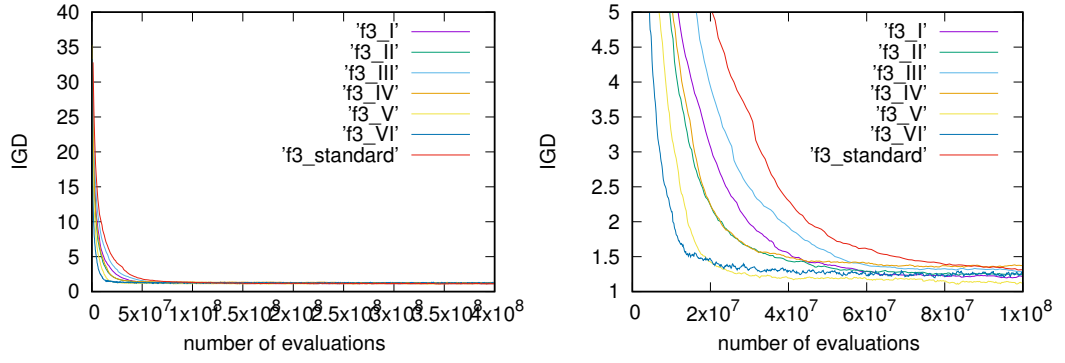


Figure 4.7: Individual effects by applying those strategies on TF3 .

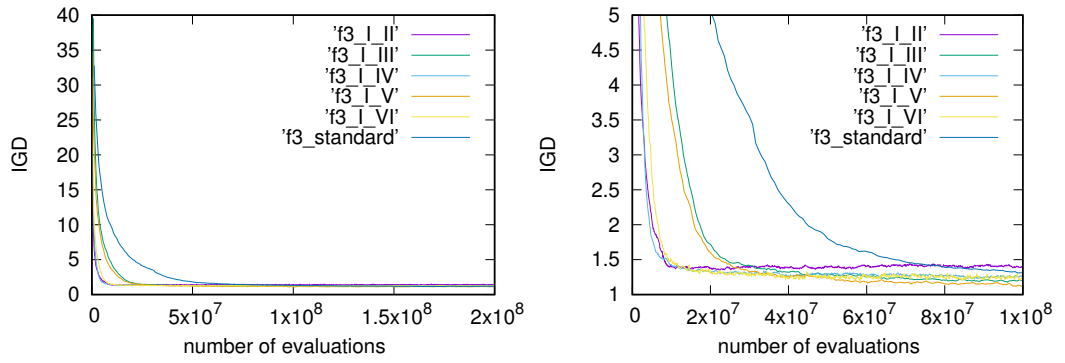


Figure 4.8: Effects of combining Strategy I with other on TF3 .

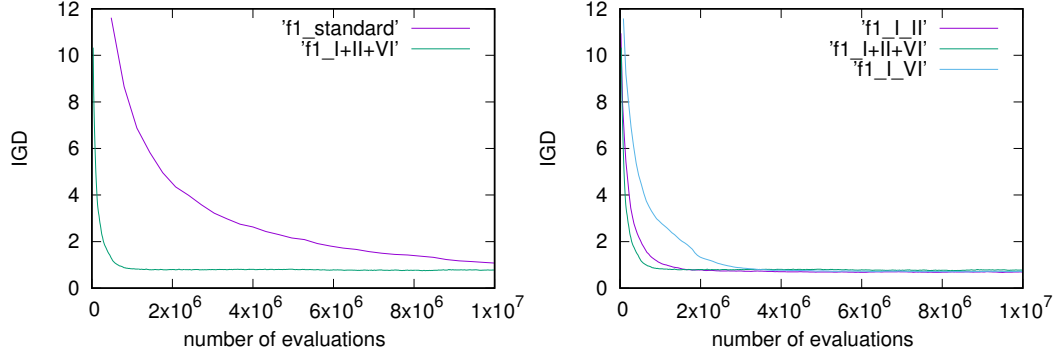


Figure 4.9: Effects of combining strategies I+II+VI on TF1 .

4.6 Test results for the combination of three strategies

We also try to combine three strategies, in this section we combine strategies I+II+VI and the results are displayed in Figure 4.9. On the left side, we compare the algorithm by applying strategies I+II+VI with baseline algorithm. It is clearly shown that the algorithm with combination of strategies I+II+VI converges much earlier than the standard algorithm, and the IGD value does not look bad. On the right side, the algorithm with combination of strategies I+II+VI has earlier convergence than the combination of strategies I+II and I+VI. Although the IGD value is slightly higher, it does not sacrifice too much.

From the test results interpretation as above, we can see that those strategies work well in general. Strategy I works well in general, it aims to filter those solutions that are worth to do complete evaluation. In the point-based algorithm, the lower level is a single objective optimisation problem that searches for a worst-case fitness for an upper level solution, as the lower level runs, the worst-case case fitness updates by finding new worst-case values. If this solution is dominated by the currently found solutions, then this solution is less likely to become a non-dominated solution and it is not worth to continue the lower level run. By aborting the lower level run earlier, the fitness evaluations would be saved.

Strategy II can reduce the fitness evaluations half, because the re-evaluation process of an upper level solution is replaced by exploiting the information in its neighbourhood. Its worst-case fitness can be improved by exploring the extensive stored information in the neighbourhood.

Strategy III tries to save fitness evaluations by skipping the re-evaluation at the end of each upper level generation. If the lower level algorithm can identify the worst-case fitness correctly, then the re-evaluation process does not improve the identified lower level solution. In this case, it is not necessary to do another re-

evaluation and its solution quality is worth to be treated as reliable once. If this upper level solution survives to the succeeding generation, its re-evaluation is not conducted once. However, this method could fail if the lower level can not find the accurate worst-case fitness or the evaluated upper level solution cannot survive to the subsequent generation.

Strategy IV attempts to reduce the number of fitness evaluations by stopping the re-evaluation process earlier if there is no significant improvement on the lower level solution quality. However, if the worst-case fitness is not identified correctly, the upper level search could be misguided towards those solutions that appear good but actually not.

Strategy V and VI adjust the lower level population size and generation size according to the tolerance levels. If the optimal solutions locate at the search region with small tolerance levels, then the fitness evaluations could be saved extensively.

In the point-based algorithm, Strategy I, II and I+II have good performance with earlier convergence and reasonable IGD values.

4.7 Summary

We suggested and compared various strategies to reduce the necessary number of fitness function evaluations in bi-level worst case optimisation, in particular when looking at the trade-off between worst case and tolerance level. We record how early the algorithms stop with different strategies (measured by how much percent of the standard evaluations). In reality, we would like to know the performance of the algorithms within a fixed number of fitness evaluations for a more straightforward comparison. The point-based algorithm with different strategies and combinations of strategies have been implemented within a fixed number of fitness evaluations.

The strategies obtained a reduction of fitness function evaluations of up to 95%, with only modest decrease in performance. This is an important step towards making bi-level worst case optimisation computationally feasible. We consider the extension of the proposed strategies to the envelope based algorithm proposed in (Lu et al. 2016), and which of those strategies can be extended to general bi-level optimisation problems. Also, a better understanding of when and why the different strategies work well would be helpful. Finally, it may be a good idea to vary the usage of these strategies over the run and, e.g., switch them off towards the end of the run.

Chapter 5

Improving the efficiency of envelope-based algorithm

In the previous chapter, we proposed a number of strategies to improve the efficiency of the point-based algorithm, where the lower level is a single objective optimisation problem. In this chapter, we extend those strategies to the envelope-based algorithm, where the lower level is a multi-objective optimisation problem. For each upper level individual x_i , the lower level optimiser aims to obtain a trade-off between the worst-case quality and its tolerance level. Because of the lower level multi-objective optimisation problem, the envelope-based algorithm requires more evaluations on the lower level. It motivates us to extend the strategies to the envelope-based algorithm to reduce the number of evaluations.

The chapter is structured as follows. We introduce how our envelope-based algorithm is formulated as a bi-level optimisation problem and the motivation to improve the efficiency of this algorithm. What follows is to describe the extension of those six strategies in Section 5.2 to reduce the necessary number of function evaluations. Empirical results will be discussed in Section 5.3.

5.1 Motivation

In the previous chapter, we demonstrated that the point-based algorithm can be formalised as a bi-level optimisation problem. The proposed strategies can improve the efficiency of the algorithm significantly. In our envelope-based algorithm, both upper and lower levels are multi-objective optimisation problems.

The problem we want to solve by envelope-based algorithm can be formulated as:

$$\max \quad f(x'^*) \quad (5.1)$$

$$\max \quad \delta(x'^*) \quad (5.2)$$

s. t.

$$x'^* \in \operatorname{argmin}\{(f(x'), \delta(x')) : \delta(x') = |x - x'|, x' \in [x - \Delta, x + \Delta]\} \quad (5.3)$$

$$x \in [x^L, x^U] \quad (5.4)$$

The upper level decision variable x lies between its lower bound x^L and upper bound x^U . For an upper level solution x , its maximum tolerance level is defined as Δ which determines the lower level search space. The lower level variable is defined as x' and it belongs to the search space determined by x and Δ . The lower level is a multi-objective minimisation problem, where its optimal lower level x'^* is a solution set rather than a single solution. The lower level returns a front represented by $\{f(x'^*), \delta(x'^*)\}$ to the upper level. The two objectives minimised in the lower level are the same as those two objectives to be maximised in the upper level.

Algorithms 3 and 4 in Chapter 3 show the pseudocode for the upper and lower level multi-objective evolutionary algorithms, respectively. Each feasible upper level solution is obtained by running a lower level multi-objective optimisation problem, which requires even more fitness evaluations than the point-based lower level. Additionally, line 11 shows the re-evaluation process for the envelope-based algorithm. The previously proposed strategies can save fitness evaluations significantly, therefore we would like to extend those strategies to the envelope-based algorithm.

5.2 Strategies to save fitness evaluations for envelope-based algorithm

In this section we describe how to adapt the strategies from Chapter 4 to the envelope-based algorithm. Applying strategy I and II in the envelope-based algorithm is not straightforward, because in the envelope-based algorithm the lower level is a multi-objective optimisation problem that returns a front to the upper level. The adaptations are explained in more detail in the following sections. Extending strategy V and VI to the envelope-based algorithm is relatively straightforward. For strategy III and IV, the quality indicator used to evaluate the lower level quality is hypervolume. Because the lower level returns a Pareto front, we apply hypervolume

as the quality indicator instead of crowding distance.

5.2.1 Strategy I: Exploit upper level information for early abortion of the lower level

Strategy I can be extended to the envelope-based algorithm. Note that the lower level algorithm of the envelope-based approach returns a front to the upper level. If all the solutions on the lower level front enter into the dominated region, the lower level front would not contribute to the upper level front and we can stop the lower level run early. If there is at least one solution on the lower level front that is above the dominated region, then this means that the lower level front could bring additional value to the upper level front and would improve it. So it is worthwhile to pursue the lower level front by running more generations.

Now we explain the relationship between lower level front and the dominated region in detail. There are three cases we consider:

1. All the solutions on the lower level front enter into the dominated region.

In this case, each solution on the lower level front is dominated by at least one solution on the dominated region. The lower level front would not contribute to the upper level front. There is no need to continue running the lower level algorithm, so we can choose to stop the lower level run prematurely.

2. All the solutions on the lower level front are above the dominated region.

In this case, no solution on the lower level front is dominated by any one solution on the dominated region. The lower level would contribute to the upper level front and improve it. We will choose to continue running the lower level algorithm.

3. A fraction of solutions on the lower level front fall into the dominated region and the rest are above it.

In this case, those solutions above the dominated region would bring additional value to the upper level front and improve it. It is worthwhile to give these solutions more lower level run generations to check whether they are really so good or if any of them will enter into the dominated region eventually.

As shown in Figure 5.1, the red points are the non-dominated solutions of the upper level that form the dominated region below the blue line. A lower level front formed by the purple points belongs to Case 1 where all the solutions on the front enter into the dominated region. The upper level solution that corresponds to this

front will not be an upper level non-dominated solution, and it is not worthwhile to continue the lower level run. A lower level front formed by the blue points is characterised as Case 3, where all the solutions on the lower level front are above the dominated region. The corresponding upper level solutions are non-dominated and improve the upper level front if they really are Pareto optimal on the lower level. If the computational budget is not yet exhausted, all the solutions on the lower level front will be given more lower level run to check Pareto optimality.

A lower level front formed by the green points is defined as Case 2 in Figure 5.1, where part of the solutions enter into dominated region and the rest are above. Solutions 2 and 3 within the dominated region will not contribute to the upper level front and not be preferred in the upper level. On the other hand, solutions 1 and 4 are above the dominated region, and possibly bring additional value to the upper level front and would be preferred in the upper level. As described in the envelope-based algorithm, the lower level is a multi-objective optimisation problem that returns a Pareto front. The standard non-dominated sorting is applied, while in this strategy we make some modifications about the first non-dominated front only in Case 2. Each solution in the first non-dominated front is given a rank 0, so each solution on the green front for Case 2 has a rank 0. Solution 2 and 3 fall into the dominated region and would not contribute to the upper level front. Solution 1 and 4 has the potential to contribute to the upper level front, and we would like to search more around these solutions. Because solutions with lower rank are preferred in evolutionary algorithm selection, this biases the search towards those points that may have a contribution to the upper level front. Therefore, solution 1 and 4 are given a rank 0, while solutions 2 and 3 are given rank 0.5. The lower level will be guided to the search the promising area to find out if there is any solutions that will contribute to the upper level front.

To apply strategy I in the envelope-based algorithm, initially generate an upper level parent population. For each upper level individual, call the lower level optimiser and it will return a lower level front that contains lower level non-dominated solutions. Combine all the fronts and find all the non-dominated solutions in the upper level to form the dominated region. The next step is to generate an upper level offspring population by applying standard evolutionary operators. For each offspring individual, we run the lower level algorithm and get all the non-dominated solutions that form the lower level front at the end of each lower level generation. Compare this lower level front with the dominated region as described above. Only the upper level offspring whose lower level front is obtained by running the lower level optimiser to the end will survive and join the union. What follows is the

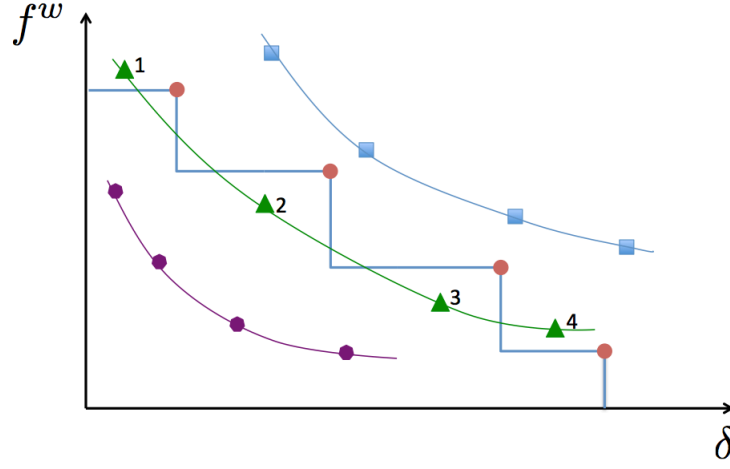


Figure 5.1: The dominated region of envelope-based algorithm

standard envelope-based upper level algorithm.

By applying this strategy, fitness evaluations are saved by stopping the lower level run earlier so as to reduce the lower level fitness evaluations. Also, this strategy focuses on most promising solutions, because if the solution potentially becomes dominated its lower level run will be stopped prematurely.

5.2.2 Strategy II: Exploit the information in the neighbourhood

We extend Strategy II to the envelope-based algorithm by exploiting information in the neighbourhood. The motivation to use this strategy is to replace the re-evaluation process. Since in our envelope-based algorithm the lower level is a multi-objective optimisation problem, we use hypervolume as a quality indicator. The hypervolume is defined as the area between the reference point and the lower level front. For instance, in Figure 5.2 solutions $A - D$ are the non-dominated solutions in the lower level that form the lower level front. The hypervolume is defined as the area between the reference point and the non-dominated solutions, as shown the area by the black dashed line. For the same reference point, a better lower level front has a larger hypervolume.

In the envelope-based algorithm framework, for each upper level solution x its maximum tolerance level Δ defines the neighbourhood of the lower level decision variable x' , where x' is restricted to $[x - \Delta, x + \Delta]$. An upper level solution corresponds to a lower level front that is the trade-off between $f(x')$ and δ , where $\delta = |x - x'|$. We would like to exploit the information in $[x - \Delta, x + \Delta]$ and create an archive to store all the information of $f(x')$. Since our algorithms are population-

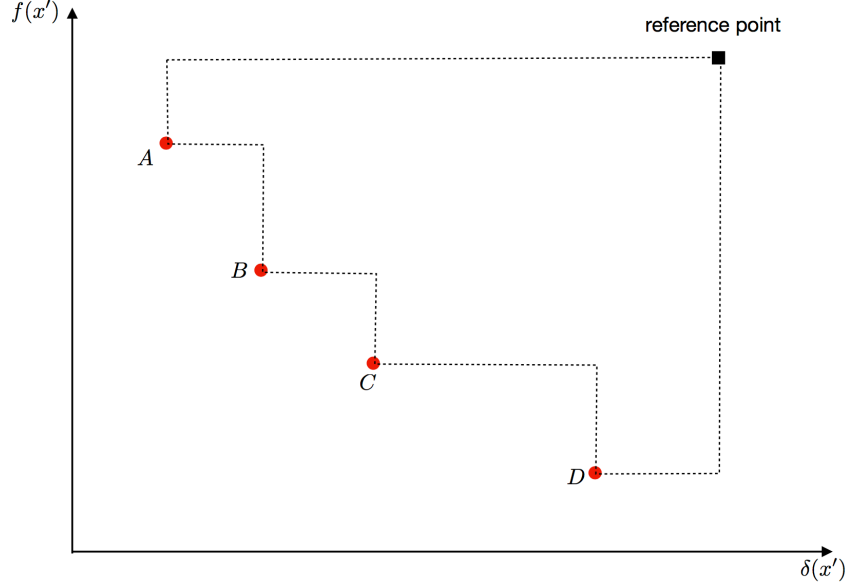


Figure 5.2: Example for hypervolume of a lower level front

based, we have a set of lower level fronts in the upper level. For the solutions on those lower level fronts, if any lower level decision variable falls into $[x - \Delta, x + \Delta]$, store $(x', f(x'))$. If any point in the archive can improve the lower level front of x , keep it and update the lower level front.

For a specific upper level solution x_1 , its lower level decision variables x'_1 are bounded to $[x_1 - \Delta_1, x_1 + \Delta_1]$. Its lower level front is formed by the lower level non-dominated solutions (e.g. $A - D$ in Figure 5.4). For other upper level solutions, for instance, x_2 , its lower level variables x'_2 are bounded to $[x_2 - \Delta_2, x_2 + \Delta_2]$. For a lower level variable of x_2 , if it is within $[x_1 - \Delta_1, x_1 + \Delta_1]$ as shown in Figure 5.3, keep $(x'_2, f(x'_2))$. The other objective of the lower level is easy to calculate as $\delta = |x_1 - x'_2|$. A lower level solution with two objectives $(f(x'_2), \delta)$ of an upper level solution x_1 will be derived. What follows is to check whether this lower level solution will improve the lower level front of x_1 . If a new lower level non-dominated solution is found, shown as solution E in Figure 5.4, the new lower level front of x_1 will be formed by $A - D$ and E . The hypervolume increases by the blue shaded area, and the lower level front will be improved by solution E . If a newly found solution dominates one of the solutions on the lower level front, for instance, solution B' dominates solution B , then the new lower level front is formed by A, B', C, D . The hypervolume would raise the blue dashed line area. The lower level front will be improved by solution B' . Therefore, both B' and E will be kept to update the lower level front.

Apply the process described above to all the other upper level solutions of the

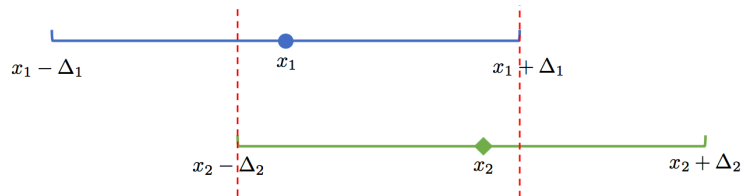


Figure 5.3: The comparisons of two upper level solutions

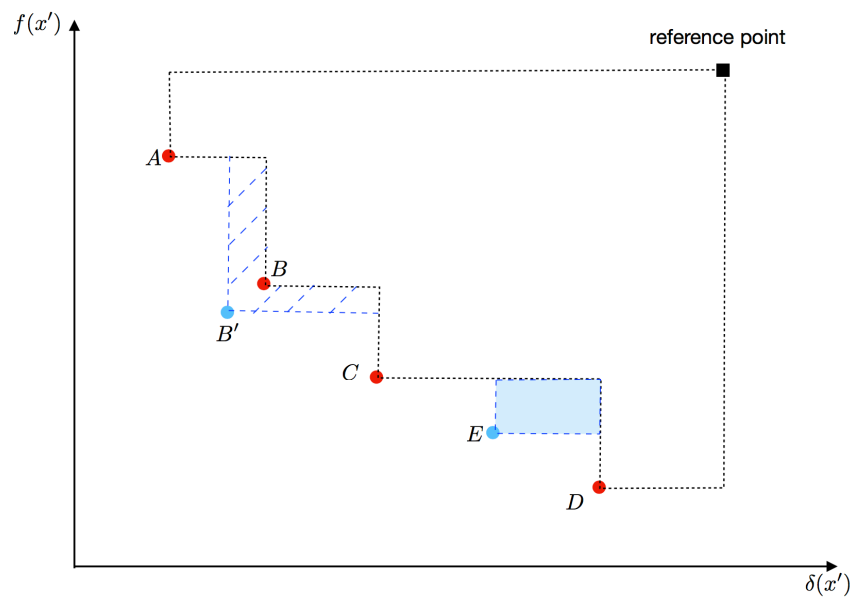


Figure 5.4: If new, better solutions are found by lower level searches of neighbouring solutions

union population (the combination of parent and offspring upper level population), an improved lower level front of x_1 will be obtained.

To use this strategy in our envelope-based algorithm. Initially, generate the upper level parent population and evaluate each individual. Non-dominated sort the parent population, generate the upper level offspring population, and evaluate each solution in the offspring. Get the union of parent and offspring population. Each upper level solution in the union has its lower level front. We apply this strategy II to the union. The lower level front of each upper level solution in the union may be improved by the rest lower level fronts. The next step is to non-dominated sort the union and get the new upper level parent population that survives to the next generation. Fitness evaluations will be reduced by half, because Strategy II replaces the re-evaluation process.

5.2.3 Strategy III: Skip re-evaluation if there is no improvement of the lower level front

For strategy III, in the re-evaluation process, we use hypervolume as the quality indicator. For each survived upper level solution, we combine the lower level fronts before and after re-evaluation to find the non-dominated solutions that form a new front. If the new front size is larger than the lower level population size, we select the number of lower level non-dominated solutions with lower level population size using crowding distance.

Compare the hypervolume of this new front with the old one (the lower level front before re-evaluation). If there is no improvement in hypervolume, this means the re-evaluation did not identify an improved lower level front, the lower level front previously found seems reliable and it is worth to be trusted once. This strategy will thus choose to skip the re-evaluation for this upper level solution in the following generation.

5.2.4 Strategy IV: Lower level smart initialisation

The re-evaluation process in the envelope-based algorithm aims to identify the lower level front correctly for each upper level solution. In the baseline algorithm described in Chapter 3, the lower level is re-started from scratch for re-evaluation. Strategy IV suggests to keep the population at the end of a lower level run in memory, and re-start the search from this population if the corresponding upper level solution is to be re-evaluated. Compared with this strategy in the point-based algorithm, because of the inherent diversity of a non-dominated front it is not likely to get stuck

in local optima. It does not make much sense to continue the lower level runs if there is not much improvement on the lower level front, and we stop the lower level run early in case the improvement over the past 5 generations was less than 0.01. The quality indicator used to evaluate the improvement of the lower level front is hypervolume.

5.2.5 Strategy V: Lower level generations adaptive to Δ

For each upper level solution, its corresponding maximum tolerance level that defines the lower level search space is given by Δ . For a small Δ , the lower level search space is small, and it requires less number of evaluations in the search process. Therefore, it makes sense that we set the lower level generations size adaptive to this maximum tolerance level. For strategy V, we will use

$$Generations = \max \left\{ 10, \left\lceil maxGen \times \frac{\Delta}{\delta^{max} - \delta^{min}} \right\rceil \right\}$$

5.2.6 Strategy VI: Lower level population size adaptive to Δ

As described above, the lower level search space for an upper level solution x is determined by Δ . For strategy VI, the lower level maximum population size is $maxPopSize$ and we define

$$popsize = \max \left\{ 2, \left\lceil maxPopSize \times \frac{\Delta}{\delta^{max} - \delta^{min}} + 1 \right\rceil \right\}$$

in case of a single decision variable, and

$$popsize = \max \left\{ 2, \left\lceil maxPopSize \times \frac{\Delta^2}{(\delta^{max} - \delta^{min})^2} + 1 \right\rceil \right\}$$

for the case of two dimensional problems.

5.3 Empirical results for envelope-based algorithm by applying simple strategies

In this section, we report on the empirical results of applying those six strategies for the envelope-based based algorithm where the lower level is a multi-objective optimisation problem. The test results are based on the test problems introduced in Chapter 2.

Table 5.1: Parameter setting for standard envelope-based algorithm on TF1 and TF2

	TF1 & TF2		TF3	
	UpperLevel	LowerLevel	UpperLevel	LowerLevel
popsiz	40	100	40	100
max generations	100	100	200	250
crossover prob.	0.9	0.9	0.9	0.9
mutation prob.	0.9	0.9	0.5	0.5

5.3.1 Parameter settings

Table 5.1 shows the parameters setting of the standard envelope-based algorithm for different test functions. For the one dimensional test functions 1 and 2, the upper level MOEA was run for 100 generations with a population size 40, and the lower level MOEA population size and number of generations are both set as 100. The crossover probability of both upper and lower level is set as 0.9. The mutation probability of both upper and lower level is set as 0.9, because both upper and lower level decision variables are one dimension. For the 2-dimensional problems, we increase the generation size to 200 in the upper level and 250 in the lower level. Both upper and lower level crossover probability is set as 0.9. Because for a 2-dimensional problem, both upper and lower level decision variables are two dimensions and we use a mutation probability of 0.5. For the performance quality measurement, we use Inverted Generational Distance (IGD). It evaluates how the algorithms converge to the true optimal solutions. Each algorithm is run for a given number of evaluations.

5.3.2 Test results and analysis

In the previous chapter, we introduced six simple strategies to save fitness evaluations for the point-based algorithm, and the main goal was to improve the efficiency of the algorithm. In this chapter, we extended those simple strategies to the envelope-based algorithm to reduce the number of evaluations. The envelope-based algorithm aims to search for upper level solutions for which the corresponding lower level front has a contribution to the upper level Pareto front. We will show the individual effects of those six methods and then look at the effects of the combinations of those methods. The convergence plots show how the IGD values decrease with the number of evaluations.

Table 5.2: Final IGD value when each of the six different strategies are used individually

Strategy.	TF1		TF2		TF3	
	Gens	IGD \pm std.err.	Gens	IGD. \pm std.err.	Gens	IGD \pm std.err.
Standard	100	0.609 ± 0.002	100	1.428 ± 0.002	200	1.276 ± 0.062
I	186	0.576 ± 0.001	170	1.361 ± 0.009	351	1.265 ± 0.016
II	200	0.601 ± 0.041	200	1.314 ± 0.054	400	1.021 ± 0.001
III	165	0.612 ± 0.001	188	1.395 ± 0.005	268	1.423 ± 0.002
IV	191	0.653 ± 0.003	191	1.429 ± 0.004	392	2.34 ± 0.021
V	179	0.653 ± 0.000	136	1.375 ± 0.000	425	1.397 ± 0.005
VI	177	0.621 ± 0.003	135	1.429 ± 0.002	783	2.830 ± 0.014

Individual effects on envelope-based algorithm

Firstly, we look at the individual effects of each strategy on different test functions. Table 5.2 shows the final IGD values of different methods by running each algorithm given a fixed number of evaluations of 8×10^7 for one dimensional test functions 1 and 2, and 4×10^8 for two dimensional test function 3. The first for each test function column shows how many generations the algorithm can run with the given number of evaluations. The last column gives the average IGD value with standard deviation over 20 runs for each method.

The convergence plot of the simple test function 1 is shown in Figure 5.5. Strategy II has the best performance with an early convergence and the lowest IGD value. What follows is Strategy I and V with earlier convergence and the IGD value does not sacrifice much. The algorithm with Strategy IV converges earlier but has the worst IGD value. Strategy III and VI have earlier convergence but the IGD value is slightly worse than the baseline algorithm after about half of the total evaluations. In general, the IGD values do not vary much, because this a simple test function with only three optimal upper level non-dominated solutions.

For the test function 2 with more peaks, the convergence plot is displayed in Figure 5.6. What is consistent is that Strategy II has much earlier convergence and lowest IGD value at about 1.3. Strategy I has slightly earlier convergence but lower IGD value than the baseline envelope-based algorithm. Strategy VI converges slower with worse IGD value from the beginning, and it converges close to the baseline algorithm after about half of the total evaluations. Strategy V does not have much earlier convergence but the IGD value lies below the baseline algorithm. Strategy III and IV have earlier convergence, the difference is that Strategy IV converges to the IGD value that is close to the baseline algorithm, while Strategy III can have a better IGD value. Strategy III does not have much earlier convergence but the IGD

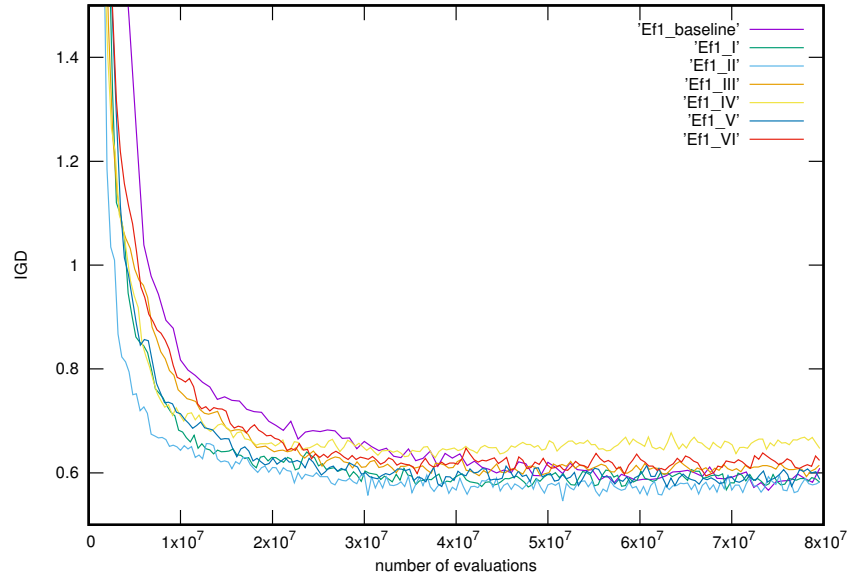


Figure 5.5: Individual effect of applying six strategies on envelope-based algorithm of TF1

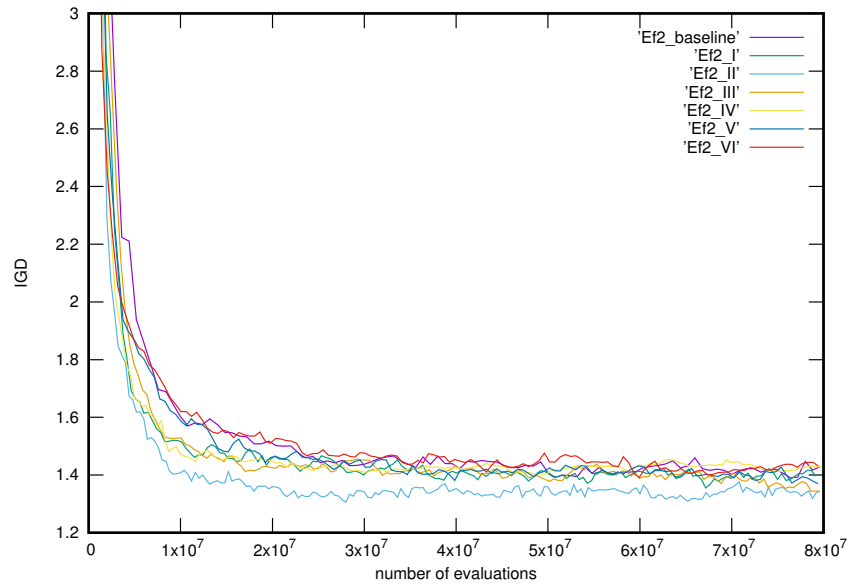


Figure 5.6: Individual effect of applying six strategies on envelope-based algorithm of TF2

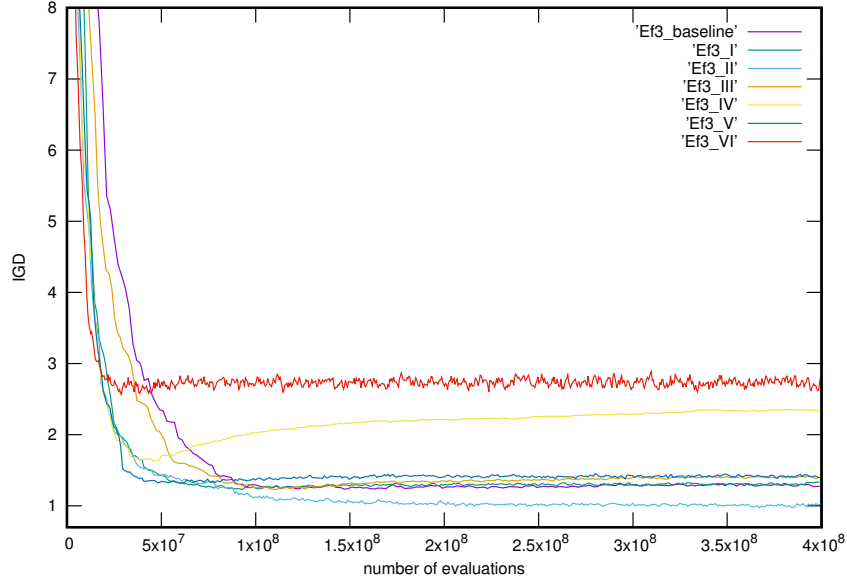


Figure 5.7: Individual effect of applying six strategies on envelope-based algorithm of TF3

value is getting better than the baseline algorithm.

For this 2-dimensional test function 3, the convergence plot is shown as Figure 5.7. As it can be seen from the figure, the standard envelope-based algorithm converges to the IGD value about 1.2. By applying strategy I, the algorithm converges much faster with a slightly worse IGD value. The aim of Strategy I is to evaluate the upper level solution fully if it is possible to be better than the current upper level non-dominated solution set. Strategy II (exploit information in the neighbourhood) performs best with a very early convergence and the best IGD values close to 1.0. Using Strategy III, the algorithm has better performance in the beginning but converges to a worse IGD value than the baseline algorithm after about 1.4×10^8 evaluations. Strategy V adjusts the lower level generation size to the tolerance levels. This strategy converges with a small number of evaluations of about 4×10^7 , and its IGD value is about 1.4 which is slightly worse than the baseline algorithm.

What should be noticed is that the algorithms with Strategy IV and VI have much worse performance than the baseline algorithm. It is clear from the convergence plot that Strategy IV has better performance than the baseline algorithm in the beginning but later the IGD value goes up to about 2.34. Because the lower level is a multi-objective minimisation problem, if the lower level does not find the true trade-off between worst-case quality and robustness, with solutions that ap-

Table 5.3: Final IGD of combining Strategy I with any of the other five strategies

Strategy	TF1		TF2		TF3	
	Gens	IGD \pm std.err.	Gens	IGD \pm std.err.	Gens	IGD. \pm std.err.
Standard	50	0.594 ± 0.051	50	1.428 ± 0.002	100	1.265 ± 0.016
I+II	1501	0.735 ± 0.009	653	3.261 ± 0.023	1660	0.981 ± 0.007
I+III	421	0.630 ± 0.002	250	1.373 ± 0.002	468	1.433 ± 0.001
I+IV	706	0.640 ± 0.001	313	1.390 ± 0.003	1589	2.090 ± 0.004
I+V	169	0.594 ± 0.003	96	1.403 ± 0.004	347	1.415 ± 0.003
I+VI	169	0.615 ± 0.002	99	1.419 ± 0.001	620	2.591 ± 0.007

pear to be better in the upper level, this will guide the upper level to find a Pareto front with solutions seemingly better than the true Pareto front. But actually these solutions are not true, so the approximate trade-off between worst-case quality and robustness obtained by applying Strategy IV will be far from the true Pareto front. This explains the importance of re-evaluation process at the end of each generation. Because Strategy IV emphasis on the early stop of the re-evaluation, if the lower level algorithm is not able to obtain an accurate trade-off, the lower level solution quality could not be improved with early abortion in the re-evaluation.

Strategy VI has the worst IGD value at about 2.83, although it converges earliest compared with the rest. This is because in the envelope-based algorithm, the lower level returns an envelope. Looking at the lower level population size setting, the lower level envelope does not provide sufficient information to the upper level. For instance, in this 2-dimensional problem, the lower level deserves more population size to have a complete envelope that returns to the upper level. A possible method to improve it is to define a better relationship between the lower level population size and tolerance levels.

Combinations of strategies on envelope-based algorithm

From the above section, we can see that Strategy I works well across those three different test functions. It aims to only evaluate the upper level solution fully if it is able to improve the current non-dominated upper level solution set. We would like to combine this method with the rest and examine if the baseline envelope-based algorithm can be improved further. Table 5.3 compares the final IGD values of different methods by combining Strategy I with others for half of the total number of evaluations.

Figure 5.8 shows the effects of the combination of Strategy I with others on TF1. In general, each combination of two strategies has better performance than the baseline algorithm. As can be seen from the figure, Strategy I+II has remarkable

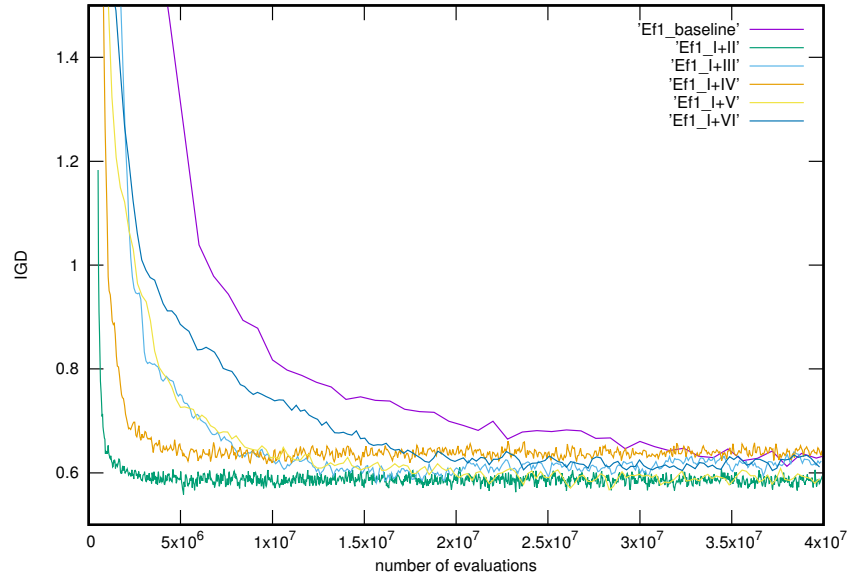


Figure 5.8: Effects of combining Strategy I with others on envelope-based algorithm of TF1

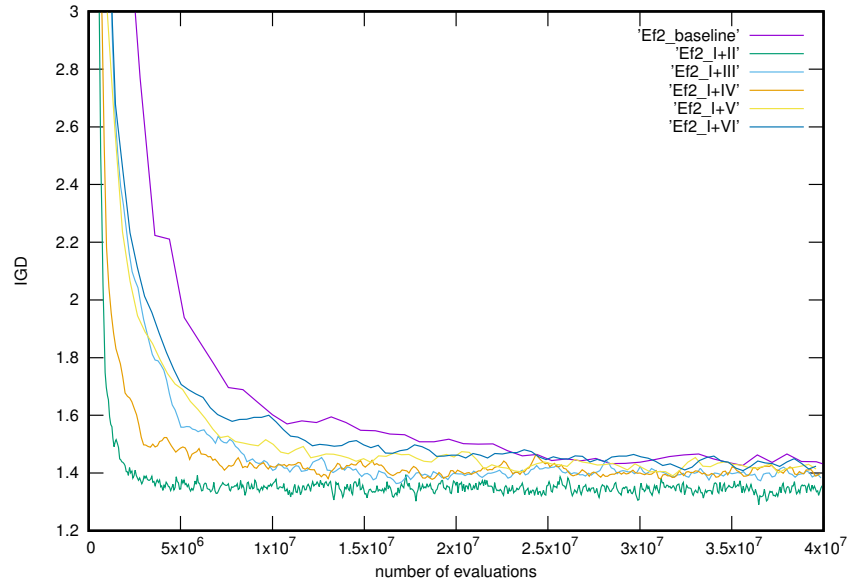


Figure 5.9: Effects of combining Strategy I with others on envelope-based algorithm of TF2

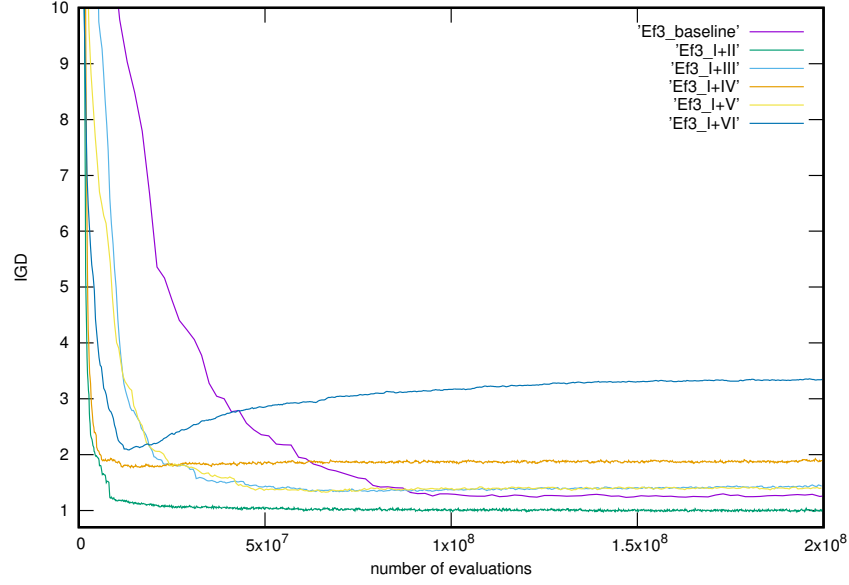


Figure 5.10: Effects of combining Strategy I with others on envelope-based algorithm of TF3

performance with the earliest convergence and the best IGD value converges to about 0.5. What should be noticed is that it converges to a good IGD value by using less than 5×10^6 evaluations, which is about 6.2% of the total number of evaluations. What follows is Strategy I+IV considering the convergence speed, but it converges to a slightly higher IGD value. Strategy I+III and Strategy I+V converge with around 1×10^7 evaluations with a good IGD value. Strategy I+VI converges earlier with a better IGD value.

The effects of combining Strategy I with others on TF2 is demonstrated in Figure 5.9. What is consistent is that Strategy I+II has the best performance with the earliest convergence and the best IGD value. Next is Strategy I+IV where the convergence is slightly slower and the IGD value is slightly higher than for Strategy I+II. Strategy I+III converges slightly slower than I+IV but the IGD value is very close. Compared to the baseline algorithm, Strategy I+V and I+VI have minor improvement.

The effects of combining Strategy I with others on envelope-based algorithm of the 2-dimensional TF3 is shown in Figure 5.10. It is clear from the figure that the combination of Strategy I+II again has the best performance. It has the earliest convergence and the best IGD value of about 1.0. What follows is Strategy I+III and I+V, which converges much earlier but with a slightly worse IGD value at about 1.433 and 1.415, respectively. Strategy I+IV converges as early as Strategy I+II but

the IGD value is much worse at about 2.090. What should be noted is that Strategy I+VI has the worst performance where the IGD goes up to about 3.350. Looking at the individual effect, Strategy VI has the worst IGD value. If the lower level front can not be identified accurately, this will influence the upper level Pareto front so as to the dominated region in Strategy I. The combination of Strategy I+VI will cause the algorithm to search wrongly.

In general, Strategy I works well across those test functions. Using Strategy II, the number of evaluations can be reduced by exactly half. From the test results analysis above, we can see that Strategy I and Strategy II have good performance across the three different test functions. Strategy I tries to stop the lower level run earlier. Because the lower level is a multi-objective minimisation problem, as the lower level optimiser runs the lower level front moves down. We still use the dominated region in the envelope-based algorithm. If the entire lower level front enters into the dominated region, there is no need to continue the lower level run. This helps to detect those upper level solutions for which it is worth to run the lower level optimisation algorithm fully.

Strategy II aims to replace the re-evaluation at the end of each upper level generation by exploiting the information in the neighbourhood. This method focuses on improving the current lower level front obtained by the lower level optimiser by checking the information in its neighbourhood, rather than searching for another lower level front by calling the lower level optimiser. Because both upper and lower level are population-based algorithms in our envelope-based framework, there is extensive information in the neighbourhood saved. This allows to find those lower level solutions that can improve the current lower level front. Therefore Strategy I+II has the best performance among all those methods, it has the earliest convergence and best IGD value.

Strategy III attempts to save fitness evaluations by skipping the re-evaluation process. It will save significant evaluations when the lower level requires a large number of evaluations. It works well if there are two conditions satisfied. On the one hand, the lower level algorithm can find the lower level front accurately so the re-evaluation process is not likely to improve the lower level solutions quality. On the other hand, the upper level solution survives to the subsequent generation. This strategy will fail if the re-evaluation always improves the lower level solution quality. It is the case that in the beginning of the algorithm run, the skip of re-evaluation is less. The number of time re-evaluation is skipped increases over the run.

Strategy IV aims to reduce the number of evaluations in the re-evaluation process. When re-evaluating a survived upper level solution, if there is not much

improvement on the lower level quality, it is not necessary to continue running the lower level algorithm. This strategy works well if the lower level algorithm can find the lower level front correctly for an upper level solution. The re-evaluation will stop quickly if there is not significant improvement on the lower level front. In this envelope-based algorithm, the lower level is a multi-objective optimisation problem and returns a trade-off. Hypervolume is used as the quality indicator to measure the lower level solution quality. In the implementation of Strategy IV, the re-evaluation will be aborted earlier if the quality improvement is no more than 0.01 within 5 generations. The potential danger is that if the lower level front is not identified accurately, the upper level algorithm search could be misguided and ends up with a worse IGD value.

Strategy V aims to adjust the lower level population size to the tolerance levels. The algorithm tends to search more in the promising region. If the algorithm searches towards solutions with small tolerance levels, the fitness evaluations could be reduced significantly.

Compared to other strategies, Strategy VI has worst performance. In our envelope-based algorithm, the lower level returns an envelope that is formed by a set of non-dominated solutions. Strategy VI makes the lower level population size adaptive to tolerance levels. For an upper level solution with a small tolerance level, its lower level population size is adjusted to be small. This may cause the problem that the lower level does not provide sufficient solutions to contribute to the upper level front, so the quality of the upper level Pareto front may be affected. If the upper level optimal solutions are the solutions with small tolerance levels, the baseline algorithm with Strategy VI could converge to a worse Pareto front.

In our envelope-based algorithm, Strategy I, II and Strategy I+II would be recommended to apply to reduce the number of evaluations. Using any of those recommended strategies, the number of evaluations can be saved extensively without sacrificing the IGD values.

5.4 Summary

Several strategies have been proposed to reduce the number of evaluations in the point-based algorithm. In this chapter, we extended those to the envelope-based algorithm. Strategy I, II and I+II work best across three test problems. In Strategy I, when evaluating an upper level solution by the lower level optimiser, we still use the dominated region to decide whether to stop the lower level run earlier. The difference is that in the envelope-based algorithm, the lower level returns a trade-off formed

by a set of non-dominated solutions. We need to check if the whole lower level front enters into the dominated region. Strategy II attempts to replace the re-evaluation and the information in the neighbourhood is exploited to improve the lower level solution quality, and this can save exactly half of the total number of evaluations. Strategy III suggested that for an upper level solution, if the re-evaluation does not improve its performance, it deserves to be considered as reliable once. Strategy IV indicates that in the re-evaluation if there is not much improvement, it makes sense to stop the re-evaluation. Both Strategy III and IV modify the re-evaluation process, they will depend on the lower level algorithm accuracy. They can save significant evaluations if the lower level algorithm can find the optimal solutions accurately. Strategy VI is about adapting the lower level population size to tolerance levels. Because the lower level is a multi-objective optimisation problem, if the lower level population size is not large enough, the upper level solution quality will be affected.

Chapter 6

Surrogate-assisted algorithms

In Chapter 3, two algorithms have been proposed to find the trade-off between worst-case quality and robustness. Both algorithms are bi-level, the first is defined as point-based algorithm in the sense that the lower level is a single objective optimisation problem. The second algorithm is defined as envelope-based algorithm, where the lower level is a multi-objective optimisation problem and returns a set of non-dominated solutions (which we call an envelope). Because both algorithms are bi-level, each upper level solution is evaluated by calling a lower level optimiser.

In bi-level optimisation, the feasible upper level decision variables are subject to a lower level optimisation problem. This is computationally expensive especially when the lower level is optimised using EAs which are population-based algorithms. In practice, it is usually desirable to obtain an approximated optimal solution within a limited computational budget. In order to improve the efficiency, in Chapter 4 we introduced and compared several strategies to save fitness evaluations for the point-based algorithm. In Chapter 5, we adapted the six simple strategies to reduce the number of fitness evaluations to the envelope-based algorithm. All those strategies are mainly based on modifying the lower level running process. In this chapter, we would like to combine EAs with surrogate models to reduce the computational cost. The motivation is to identify which upper solution is potentially a good one that we should evaluate accurately by calling a lower level optimiser. For those solutions that are not deemed promising by the surrogate model, we save the computational cost of running the lower level optimiser.

Usually, in surrogate-assisted algorithms, surrogate models are applied to approximate the function values in order to save fitness evaluations, especially for those functions that are expensive to evaluate. In this chapter, for a decision variable and its tolerance level we apply a surrogate model to learn worst-case fitness rather

than the actual fitness. The idea is to help identify those promising upper level solutions that are worth to be evaluated appropriately using a mechanism called “pre-selection” (Allmendinger et al. 2017). Put in another way, we evaluate all upper level solutions using the surrogate model, identify the promising solutions based on the approximation values and evaluate them more accurately by running the lower level optimiser. Compared to the baseline point-based and envelope-based multi-objective evolutionary algorithm (MOEA), computational cost can be saved by reducing the number of lower level optimiser calls. What is different from the previous strategies is that we identify the promising upper level solutions without running the lower level optimiser. In our problem, we maximise the robustness and worst-case quality where the tolerance level δ defines the robustness. There is no need to evaluate δ , as it is a decision variable, so the surrogate model is only used to approximate the other objective worst-case quality f^w . We use Gaussian processes to approximate the worst-case fitness, and this aims to replace the lower level optimiser.

In the point-based algorithm, we would like to build a Gaussian process (GP) from the current evaluated solutions to learn the worst-case fitness f^w for each upper level solution $x_u = \{x, \delta\}$. On the other hand, in the envelope-based algorithm, for each upper level solution $x_u = \{x\}$ we approximate a set of worst-case values for different tolerance levels based on each GP model. Note that in the envelope-based algorithm, we approximate the lower level trade-off to replace the lower level multi-objective optimiser. Given that most fitness evaluations occur at the lower level, this strategy should allow us to save a substantial fraction of the total number of necessary fitness evaluations.

As for the model management, it classified the techniques for surrogate management into individual-based, generation-based and population-based (Jin 2011). The individual-based technique means that some of the individuals at each generation is evaluated by their real fitness evaluations. In the generation-based surrogate model management, it means that in some of the generations surrogate models are applied to fitness evaluations. In the other generations, the real fitness evaluations are used. As for the population-based surrogate model management, there are more than one sub-population, and each sub-population applies its own surrogate to do fitness evaluations. Our model management technique belongs to the individual-based model management, since we decide for each individual separately whether it should be evaluated fully or not. Our method is similar to the pre-selection strategy, where all the individuals in the offspring population are evaluated by a surrogate model (the GP model in our work), and then only the promising solutions are kept

and will be evaluated by the real fitness evaluation. However, where pre-selection usually fully evaluates a fixed number of solutions, in our case the number of solutions regarded as promising is variable.

The structure of this chapter is as follows. We introduce the related work and background at first. Then the GP of the problem is explained. In the following section, we describe the surrogate-assisted point-based and envelope-based algorithms frameworks. Compared to the baseline algorithms, the results show that this allows to reduce the number of evaluations significantly. Finally, we make conclusions.

6.1 Background and related work

Generally, GPs are used to build a surrogate model to approximate the fitness of a solution instead of the actual more expensive evaluation. It is usually combined with EAs to accelerate the algorithm (Jin 2011), especially for those objective functions that are expensive to evaluate (Ong et al. 2003) (Zhou et al. 2004) (Tabatabaei et al. 2015). Combining surrogate models with MOEAs has been used to save computational costs and improve the efficiency of algorithms, for instance, in engineering design (Ray & Smith 2006). Marzat et al. (2013) address worst-case global optimisation by considering black-box optimisation problems with uncertainty in continuous control variables and environmental variables. They relax the difficulty of the problem by assuming the continuous search space is finite and combining the EA with Kriging-based optimisation. A surrogate model is used to approximate the fitness within a certain neighbourhood to find robust design for problems with uncertainty in decision space (Ray & Smith 2006).

For bi-level optimisation problems, a significant number of fitness evaluations are required because each feasible upper level solution is obtained by solving a lower level optimisation problem, when the lower level is not solved by an exact method. Surrogate models are usually used to approximate the lower level evaluation. Combining surrogate models with differential evolutions (DE) is used to solve bi-level programming (Angelo et al. 2014). In the lower level, a surrogate model based on a number of nearest neighbour evaluated points is proposed to replace the lower level DE process. Zhou et al. (2007) apply a data-parallel GP based global surrogate model to identify those promising solutions that are worth to conduct an accurate evaluation. Firstly, surrogate models are used to approximate the fitness of solutions. Then, those solutions having good fitness based on the approximation are evaluated by a memetic search.

For minmax problems, surrogate models have been applied to approximate

the lower level evaluation. A max-min surrogate-assisted evolutionary algorithm is introduced in (Ong et al. 2006) that considers the worst-case performance for different tolerance levels in decision variables. Surrogate models are used to do fitness approximations so as to reduce fitness evaluations when dealing with problems with high dimensions. The worst-case quality within this range is based on the approximated function values. Zhou & Zhang (2010) propose a surrogate-assisted evolutionary algorithm to solve minmax optimisation problems. They apply GP to build a surrogate model that represents the mapping from decision space to objective space, and only the best individuals based on surrogate models are evaluated using the true objective values. In our case, we approximate the robust function value (the worst-case fitness).

6.2 Gaussian processes

Generally, a Gaussian process is defined as a collection of random variables, any finite number of which have a joint Gaussian distribution (Rasmussen & Williams 2006). A Gaussian process of $f(x)$ can be defined by its mean function $m(x)$ and covariance function $k(x, x')$, where

$$\begin{aligned} m(x) &= E[f(x)] \\ k(x, x') &= E[(f(x) - m(x))(f(x') - m(x')))] \end{aligned}$$

The Gaussian process can be written as

$$f(x) \sim GP(m(x), k(x, x'))$$

In our problem, GP is used to approximate the worst-case fitness $f^w(x, \delta)$. The worst-case quality is obtained by running the lower level optimisation problem, and there is no guarantee that the lower level optimiser will find the true worst-case. Because the true worst-case quality can be worse than the value obtained by the lower level algorithm. We still try to apply GP to approximate the worst-case quality. The random variables represent the values of worst-case fitness.

As described above, the worst-case fitness for each upper level solution is obtained by the lower level optimisation and it has noise represented by ε . In the following applications of GPs we make the assumption that ε is normally distributed. Given the training data that contains a number of observations of inputs (x, δ) and outputs $f^w(x, \delta)$, we wish to make predications for new inputs that we have not

seen in the training set.

6.3 Surrogate-assisted point-based algorithm

In Chapter 3, we proposed a point-based algorithm to search for the trade-off between worst-case quality and robustness. Chapter 4 suggested several simple strategies to improve the efficiency of the point-based algorithm. In general bi-level algorithms, each upper level solution is evaluated by calling a lower level optimiser. In this section, we would like to approximate the robust function, which is the worst-case fitness in the point-based algorithm. This method aims to replace running the lower level single objective optimisation problem in some cases.

6.3.1 Surrogate-assisted point-based algorithm framework

We apply MOEAs in the upper level and single objective EAs in the lower level in our baseline point-based algorithm. The upper and lower level algorithms have been described in Chapter 3. In the surrogate-assisted point-based algorithm framework, for an upper level solution which contains an input variable x and its tolerance level δ , we apply a surrogate model to approximate its worst-case fitness instead of running a lower level optimiser. GP is used to identify those promising solutions in the upper level that are worth to be evaluated accurately. Compared to the baseline algorithm proposed in Chapter 3, this can reduce the number of times to call the lower level optimiser and reduce the computational cost significantly. The surrogate-assisted point-based algorithm is shown as Algorithm 5.

First, an archive S_u is used to store all the evaluated upper level solutions. The archive contains input designs (x_i, δ_i) and their worst-case fitnesses, and it will be our training dataset that is used to build GP models in subsequent iterations. Initially, we generate the parent population randomly or by applying some techniques such as Latin hypercube sampling (LHS). Each of these solutions is evaluated by calling the lower level optimiser. What follows is to non-dominate sort the parent population and generate the offspring population.

For every newly generated upper level offspring solution, we construct a local GP model. The mean and standard deviation of the worst-case quality will be predicted based on the constructed GP. If the solution is considered as “good” by looking at the estimated value, it is worth to be evaluated accurately by calling the lower level algorithm. Otherwise, this solution will most likely be dominated and is not likely to be preferred.

Then the algorithm combines parent and offspring population and performs a non-dominated sorting process to get the new parent population that survives to the next generation. The last step is to do the re-evaluation process. At each generation, the archive will be updated by adding some newly evaluated upper level solutions. Repeat the process as described above until the stopping criterion is satisfied. Because the archive is updated at the end of each generation by adding the newly evaluated upper level solutions, the accuracy of GP models will improve over time. An optimal Pareto front between the worst-case fitness and robustness will be obtained with a small number of evaluations.

Algorithm 5 Pseudocode for upper level MOEA

```

1: Create an archive  $S_u$  to store all evaluated upper level solutions.
2: Generate parent population  $P(x, \delta)$ 
3: Call lowerEA to evaluate each solution in  $P$ 
4: for  $j=1$  to  $g$  do ▷  $g$  is number of generations
5:   Non-dominate sort  $P$ 
6:   Generate offspring population  $O$ 
7:   Build a GP for each solution in  $O$ 
8:   Check for each solution whether to use the GP estimate, otherwise call the
   lower level optimiser
9:   Get the union population  $U = P \cup O$ 
10:  Non-dominate sort  $U$ 
11:  Form the next generation parent population  $P$ 
12:  Re-evaluate the new next generation population  $P$ 
13:  Update the archive by adding newly evaluated upper level solutions
14: end for

```

6.3.2 Re-evaluation in surrogate-assisted point-based algorithm

In our algorithm design, the lower level is based on heuristics. In order to improve the accuracy of the worst-case quality of each upper level solution, we would like to conduct the re-evaluation process for the survived population at the end of each generation.

Moreover, the evaluated upper level solutions stored in the archive are used to build GP. The accuracy of the worst-case fitness of those points in the archive is also of importance. If the worst-case fitness of a solution in the archive is over-estimated, the prediction value tends to be higher and this will increase the chance to call the lower level optimiser. Note that if a solution has been re-evaluated and a more accurate worst-case value has been found, the new worst-case value replaces the previously stored worst-case quality also in the archive.

6.3.3 Identity promising upper level solutions

This section explains how to identify the upper level solutions that should be evaluated by running the lower level optimiser. Usually, the variance of the prediction measures the accuracy of the prediction. Smaller variance indicates that the prediction is more accurate, so the prediction mean with smaller variance is preferred. On the other hand, for larger variance the estimate has a larger confidence interval and its fitness estimate is deemed less reliable. What we do is actually in line with what others do in the single-objective case: try to estimate whether a solution would have a chance to survive, good and uncertain solutions are more in need of being re-evaluated. Smaller variance indicates that the prediction is more reliable, so the solutions with smaller prediction variance would be preferred. However, this criterion does not fit our problem. As shown in Figure 6.1, solutions A-C are the current non-dominated upper level solutions that form the dominated region below the line. Whenever we generate a new upper level solution, we use GP to predict its worst-case fitness. For solutions D with larger variance and E with smaller variance, solution E has smaller confidence interval. In our framework, we maximise the robustness and worst-case quality at the upper level. Solution E with its upper confidence level within the dominated region will be dominated anyway even with a smaller variance. Solution D with upper bound of prediction is above the dominated region, it is possible to become non-dominated. In this case, we would like to evaluate it by calling the lower level optimiser to get its actual worst-case quality, and figure out whether it becomes non-dominated. If it enters into the dominated region, it will be dominated and discarded. If it goes above the dominated region, it will improve the current non-dominated upper level solution set. The current non-dominated region will be updated as well.

From (Rasmussen & Williams 2006), the upper bound of a prediction could be described as:

$$\text{mean} + \alpha * \text{standard deviation}$$

The parameter can be varying, if it is set high, then the upper confidence bound is likely to be above the dominated region. In this case, the lower level optimiser would be called more often, and the fitness evaluations would not be saved significantly. On the other hand, if the parameter is set lower, then the upper confidence bound tends to entering into the dominated region. This will make the lower level optimiser is less called, where fitness evaluations could be saved much. The potential danger could be that there is no sufficient information to construct GP model.

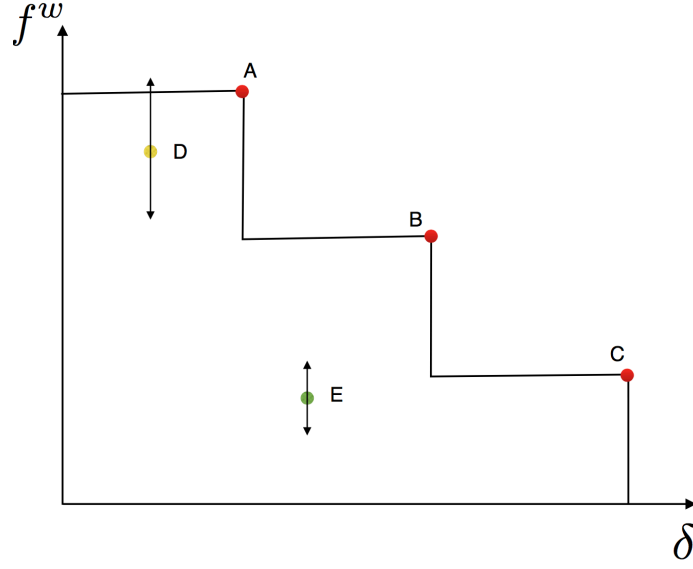


Figure 6.1: Prediction with upper limit

The 95% confidence interval for the Gaussian process model prediction is mean ± 2 standard deviation. In our problem, the upper level is a multi-objective maximisation problem where we maximise the worst-case quality, so we care about the upper confidence bound of each prediction that is defined as

$$\text{upper confidence bound} = \text{mean} + 2 * \text{standard deviation}$$

At each generation, we can find the non-dominated solutions that form the current dominated region. If the upper limit reaches outside the dominated region, the solution is regarded as a potentially non-dominated solution in the upper level. We would like to run the lower level to get the actual worst-case fitness to know if it is better than the current non-dominated solutions. If the upper limit is within the dominated region, the solution is very likely to be dominated by at least one of the current non-dominated solutions, we use the mean as predicted worst-case quality for this solution. This method can make sure all the non-dominated solutions are based on actual fitness evaluations, because those potential non-dominated solutions are evaluated by calling the lower level optimiser. Hence, the dominated region is reliable at each generation.

6.3.4 Building Gaussian process model

Let $D = \{(x_i, \delta_i), f^w(x_i, \delta_i)\}, i = 1, 2, \dots$ be the training dataset stored in an archive, which contains the evaluated upper level solutions. Each upper level solution con-

tains variables (x, δ) and its corresponding worst-case fitness f^w . The lower level is a single objective optimisation problem which searches for its worst-case quality in the neighbourhood. Then for each newly generated upper level solution, select a set of data points from the dataset to construct a GP model and predict its worst-case fitness $f^w(x_i, \delta_i)$.

In our paper, we use the squared exponential covariance function that is defined as:

$$k(x, x') = \exp\left\{-\frac{1}{2l^2}(x - x')(x - x')^T\right\}$$

where x and x' are two input designs for the problem, and parameter l defines the characteristic length-scale.

Create an archive S_u that stores all evaluated upper level solutions, and this archive will be updated at each upper level generation.

Now we describe the GP model construction:

- Step 1: For each upper level solution i , define a neighbourhood with radius r .
- Step 2: Select a number of N_i closest points from the neighbourhood.
- Step 3: Use the selected training points N_i to build local GP and predict the mean and variance of this solution.

Step 1 to 2 describe how to select the training dataset to build GP for each solution. If the information in the neighbourhood is few, for instance, the number of points in the neighbourhood is less than N_i , the GP model with less information is not likely to be reliable. In this case, we use the lower level optimiser to get its actual worst-case quality for the upper level solution. The next step is check the upper bound of each prediction.

6.4 Surrogate-assisted envelope-based algorithm

In this section, we consider the problem with multiple objectives in the lower level. For a bi-level optimisation problem, if the lower level is multi-objective it will return a number of non-dominated solutions to the upper level. This is computationally expensive which motivates us to reduce the times to call the lower level. In our problem, for each upper level solution, the lower level will return a trade-off between the worst-case fitness with respect to different tolerance levels. We use Gaussian Processes to learn the trade-off between worst-case fitness and robustness for an upper level solution x_i . The difference compared to our previously proposed strategies is that we apply GP to learn the lower level trade-off in order to find those solutions

that potentially will contribute to the upper level front. For these “good solutions”, we can call the lower level to find the actual trade-off between worst-case quality and tolerance levels.

In the surrogate-assisted point-based algorithm, each GP model is constructed to predict a single worst-case fitness. Whereas in the surrogate-assisted envelope-based algorithm, each GP model is used to predict a set of worst-case values for an upper level solution with different tolerance levels. For a fixed upper level solution, its tolerance level is bounded to a certain range. The worst-case fitness does not go up as increasing the tolerance level. For this fixed upper level solution where only its tolerance level changes, we would like to learn the relationship between its worst-case fitness and tolerance level.

6.4.1 Surrogate-Assisted envelope-based algorithm framework

In our envelope-based algorithm, both upper and lower levels are multi-objective optimisation problems and it has been described in Chapter 3. The idea behind the surrogate-assisted envelope-based algorithm is to construct a GP model for each upper level solution to predict its lower level front, and this aims to replace the lower level optimiser which is more computationally expensive in the case of multi-objective.

The surrogate-assisted envelope-based algorithm is described as Algorithm 6. An archive A_u is created to store all evaluated upper level solutions. Initially, we generate an upper level parent population, and evaluate each solution by calling the lower level MOEA. Non-dominate sort the parent population and generate an offspring population. For each solution in the offspring population, select solutions from the archive to build a GP model. Use the constructed GP to predict the lower level front of this offspring solution. What follows is to decide whether to use the predicted lower level front, otherwise call the lower level optimiser to evaluate this offspring solution. Combine the newly evaluated offspring population and the parent population to form the union population, and then non-dominate sort the union population to get the next generation upper level parent population. At the end of each generation, re-evaluate the survived parent population and update the archive by adding newly evaluated offspring solutions. Repeat the described process until the stopping criterion is satisfied. The upper level front will be updated at each generation and evolves towards to the true Pareto front. An upper level Pareto front between the worst-case quality and robustness will be obtained.

In the archive, each upper level solution has a lower level front that is a trade-off between the worst-case quality and its tolerance levels. The lower level front is

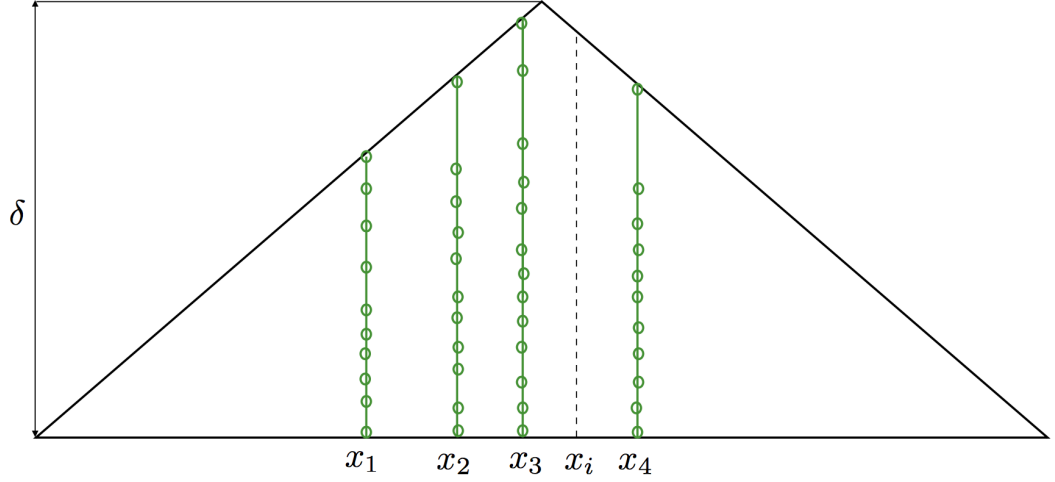


Figure 6.2: The concept of learning worst-case fitness with robustness at lower level

formed by a set of lower level non-dominated solutions. This can be explained by Figure 6.2. For each upper level solution, its lower level front is represented by (x, δ) shown as the green points in the figure that maps to (f^w, δ) in the objective space. For a newly generated upper level solution, find the closest point on either side. Now we have two lower level fronts, and the next step is to select the training data set from those two fronts to build a GP. Use this constructed GP model to predict a set of worst-case values for this upper level solution with different tolerance levels. For instance, for an upper solution x_i , the closest solutions are x_3 and x_4 from each side. The data points on the lower level front of x_3 and x_4 would be used to build the GP model for x_i .

For the new lower level predicted trade-off, we still check the upper bound of each prediction. If it is dominated by the current upper level front, the lower level front will fall into the dominated region anyway, we use the mean as prediction. If there are any points that are non-dominated, this indicates that the lower level front has potential to improve the upper level Pareto front, therefore we would like to call the lower level run to get the actual lower level front to figure out if it will improve the current upper level front.

6.4.2 Re-evaluation in surrogate-assisted envelope-based algorithm

In this surrogate-assisted envelope-based algorithm, re-evaluation process is conducted. On the one hand, it makes the lower level front quality more reliable. This is important because the lower level front contributes to the upper level front. On

Algorithm 6 Pseudocode for upper level MOEA

```
1: Generate parent population  $P(x)$ 
2: Call lowerMOEA to evaluate each individual in  $P$  to get its lower level front
3: Create an archive  $A_u$  to store all upper level solutions
4: for  $j=1$  to  $g$  do ▷  $g$  is number of generations
5:   Non-dominate sort  $P$ 
6:   Generate offspring population  $O$ 
7:   Build GP to predict for each individual with different tolerance levels
8:   Check whether to use the prediction
9:   Get the union population  $U = P \cup O$ 
10:  Non-dominate sort  $U$ 
11:  Select individuals to form the next generation parent population  $P$ 
12:  Call lowerMOEA to re-evaluate the new next generation population  $P$ 
13:  Update the archive by adding newly evaluated offspring
14: end for
```

the other hand, we use the evaluated upper level solutions to predict the lower level trade-off of the newly generated upper level solutions. The accuracy of the lower level front will influence the prediction accuracy.

6.4.3 Identify “good” upper level solutions

In this section, we explain how to identify “good” upper level solutions that should be evaluated by calling the lower level MOEA to get its actual lower level front. Those upper level solutions that are able to bring additional values to the upper level front would be considered as “good”. In the point-based algorithm, we check the upper bound of each prediction. In the envelope-based algorithm, we also check the upper bound of each prediction, where the upper bound is defined as

$$\text{upper bound} = \text{mean} + 2 * \text{standard deviation}$$

In our envelope-based algorithm, the upper level maximises the robustness and worst-case quality while the lower level minimises these two objectives. Initially, a set of non-dominated upper level solutions represented by red points in Figure 6.3 will form the dominated region that is the area below and left to the upper level front. We compare the upper bound of the predicted lower level front with the current dominated region. For the predicted lower level front, if all the upper bounds enter into the dominated region, the lower level front would be unlikely to contribute to the upper level front, so the corresponding upper level solution is considered as “bad”

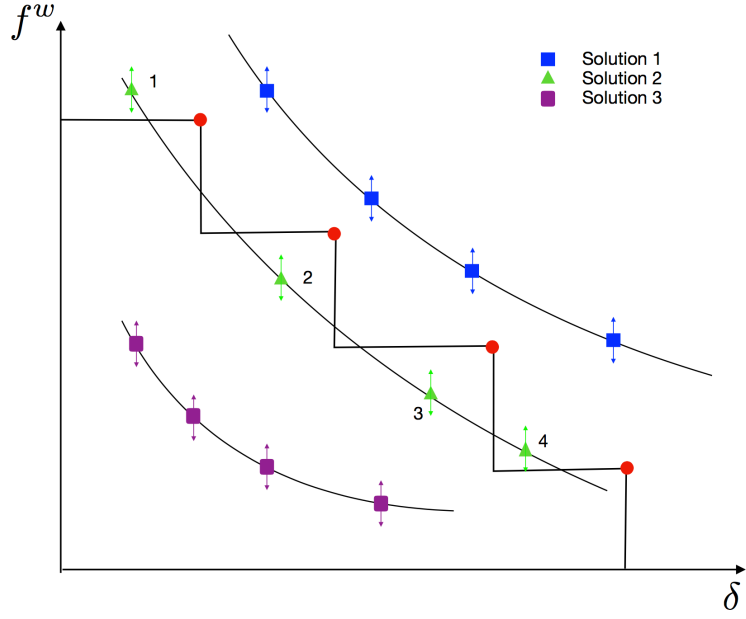


Figure 6.3: Identify promising upper level solution by GP model

and it is not worthwhile to run the lower level optimiser to get its actual lower level front. If there is more than one solution on the predicted lower level front, its upper bound goes beyond the dominated region, the lower level would bring additional value to the upper level front and improve it. In this case, we would run the lower level optimiser to get the actual lower level front and see whether it really improves the upper level front.

It is clear from Figure 6.3 that the predicted lower level front formed by the purple points would not contribute to the upper level front, as each prediction upper bound enters into the dominated region. The blue predicted lower front has potential to improve the upper level front as each prediction upper bound is above the dominated region. For the green lower level front, the prediction upper bound of solution 1 and 4 are above the dominated region, these points might bring additional values to the upper level front. For the solutions with blue and green predicted lower level front, they will be evaluated by calling the lower level optimiser.

6.4.4 Gaussian Process model for envelope-based algorithm

In the surrogate-assisted point-based algorithm, a GP model is constructed to predict a single worst-case fitness for an upper level solution. In the envelope-based algorithm, the lower level is a multi-objective optimisation problem. What is different from the point-based algorithm is that we predict the lower level front using

a constructed GP model. The covariance function is also squared exponential and has been described in surrogate-assisted point-based algorithm.

Let $D = \{x_i, (f^w, \delta)_i\}, i = 1, 2, \dots$ be the training dataset stored in an archive, which consists all the evaluated upper level solutions. For each upper level solution, its maximum tolerance level is defined as Δ . Here the upper level only contains variables x_i , and it maps to its lower level front $(f^w, \delta)_i$ that is a trade-off between worst-case quality and its different tolerance levels. Each point on the lower level front can be represented by $(f^w(x_i, \delta_{ij}), \delta_{ij}), j = 1, 2, \dots, m$, where δ_{ij} is a set of different tolerance levels of x_i that bounded to $[0, \Delta_i]$. Then for each newly generated upper level solution, select a number of data points from D , each solution in D has a lower level front that consists of a set of (f^w, δ) , combine the lower level fronts of the selected upper level solutions to build a GP model. For this newly generated upper level solution, its maximum tolerance level is defined as Δ . Now we equally generate a set of points within $[0, \Delta]$, and for each tolerance level use the created GP model to predict its worst-case quality. We will get a predicted lower level front that is a trade-off between the worst-case quality and tolerance levels.

The archive A_u is used to store all the evaluated upper level solutions, and this archive will be updated at each upper level generation.

Now we describe how to build a GP model to predict its lower level front for an upper level solution x_i :

- Step 1: For each upper level solution i , find its closest upper level solutions on either side represented as x_1 and x_2 from the archive.
- Step 2: For those two selected upper level solutions, there are two lower level fronts represented by $\{(f^w(x_1, \delta_{11}), \delta_{11}), (f^w(x_1, \delta_{12}), \delta_{12}), \dots, (f^w(x_1, \delta_{1m}), \delta_{1m})\}$ and $\{(f^w(x_2, \delta_{21}), \delta_{21}), (f^w(x_2, \delta_{22}), \delta_{22}), \dots, (f^w(x_2, \delta_{2m}), \delta_{2m})\}$, where δ_{1j} and δ_{2j} are bounded to $[0, \Delta_1]$ and $[0, \Delta_2]$.
- Step 3: Select data points from those two fronts and build GP models.
- Step 4: Predict $f^w(x_i, \delta_{ij})$ for each specific tolerance level, where δ_{ij} is a number of tolerance levels uniformly generated from $[0, \Delta_i]$.

Step 1 indicates that if there is no solution on either side, the upper level solution will be evaluated by the lower level optimiser.

The predicted lower level front for an upper level solution will be obtained. What is next is to compare this predicted lower level trade-off with the current upper level non-dominated solutions, and decide whether to evaluate this upper level solution by calling the lower level optimiser.

Table 6.1: Parameter settings for standard point-based MOEA

	TF1,TF2		TF3	
	upper level	lower level	upper level	lower level
popsize	100	40	100	40
max generations	250	40	500	100
crossover prob.	0.9	0.9	0.9	0.9
mutation prob.	0.5	1.0	0.4	0.5

6.5 Empirical evaluation and analysis

In this section, we empirically compare the proposed two surrogate-assisted algorithms using simple one and two dimensional test functions for which we can easily derive the optimum, and a test function taken from the literature. These test functions have been described in Chapter 2. We look at the effects of combining GP with the point-based algorithm first, and then examine how GP can accelerate the envelope-based algorithm.

6.5.1 Empirical evaluation of surrogate-assisted point-based algorithm

Parameter settings

The parameter setting for the baseline point-based MOEA is shown in Table 6.1. We identified suitable parameter settings for each algorithm by some preliminary test runs. The number of solutions in the Pareto front approximation corresponds to the number of individuals in the upper-level MOEA. Clearly, the upper level population size of the point-based MOEA needs to be larger. The algorithms on both levels use binary tournament for mating selection, arithmetic crossover, and Gaussian mutation. For the 2-dimensional test function 3, we use SBX crossover.

In the surrogate-assisted point-based algorithm, we define the neighbourhood radius r with a distance 1.0. For one dimensional test functions TF1 and TF2 the GP model has to map a two dimensional (x, δ) to its worst-case fitness f^w , we select 30 points as the training data set to construct GP models. For two dimensional test functions TF3, the upper level decision variables are three dimensional and we use 45 points as the training data set. For the GP training, we use R package laGP, because it allows to build a local approximation model to predict at a single input location. The length-scale is obtained by maximising the marginal likelihood. The solution found to this depends on the starting values, and this package allows to give initial value of the hyperparameters. It implements the functions that can provide

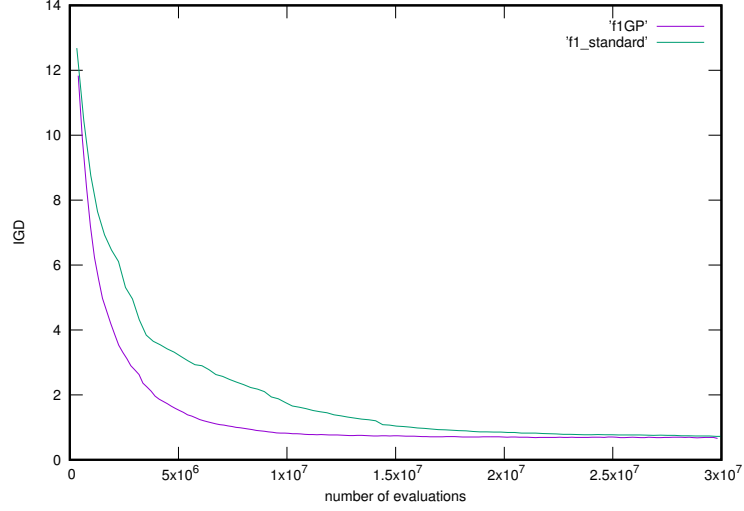


Figure 6.4: The convergence plot of surrogate-assisted point-based algorithm on TF1

good initial values to a correlation function for a GP model. It starts searching from the closest neighbour and find the required number of points that are most relevant to make predictions at the reference point.

Test results and analysis of point-based surrogate-assisted algorithm

We use Inverse Generational Distance (IGD) to evaluate the performance, and the reason has been explained in Chapter 2. Each approach proposed above is run and averaged over 20 runs.

In the surrogate-assisted point-based algorithm, the Gaussian process helps to identify the promising upper level solutions. Ideally, only the promising solutions will be evaluated by calling the lower level optimiser. The bad solutions based on GP will not be evaluated.

The convergence plots of the surrogate-assisted point-based MOEA is shown in Figure 6.4 for TF1, Figure 6.5 for TF2 and Figure 6.6 for TF3. It is clearly seen from the plots that the surrogate-assisted point-based MOEA converges earlier with a better IGD.

Moreover, we look at the archive where it stores all the evaluated upper level solutions, shown as Figure 6.7, 6.8 and 6.9. If an upper level solution is evaluated by calling the lower level optimiser, we store this upper level solution in the archive. It can help to look at the promising solutions identified by the algorithm. From Figure 6.7, we can see that the solutions converge to three solutions $x_1 = 2.5$ with δ

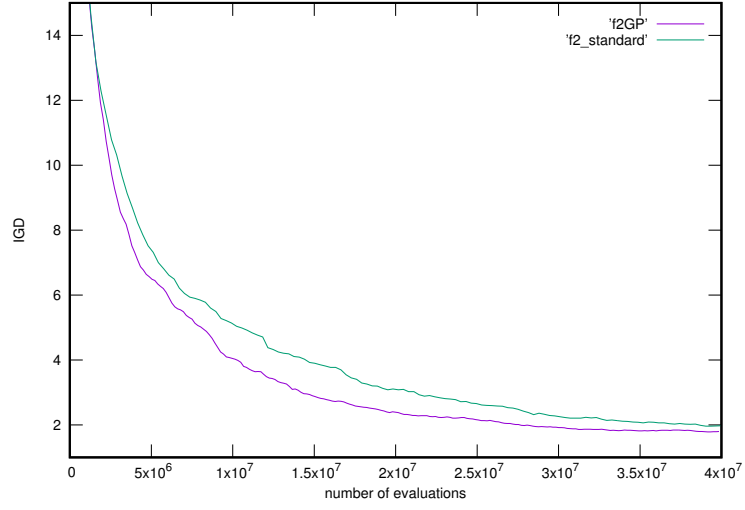


Figure 6.5: The convergence plot of surrogate-assisted point-based algorithm on TF2

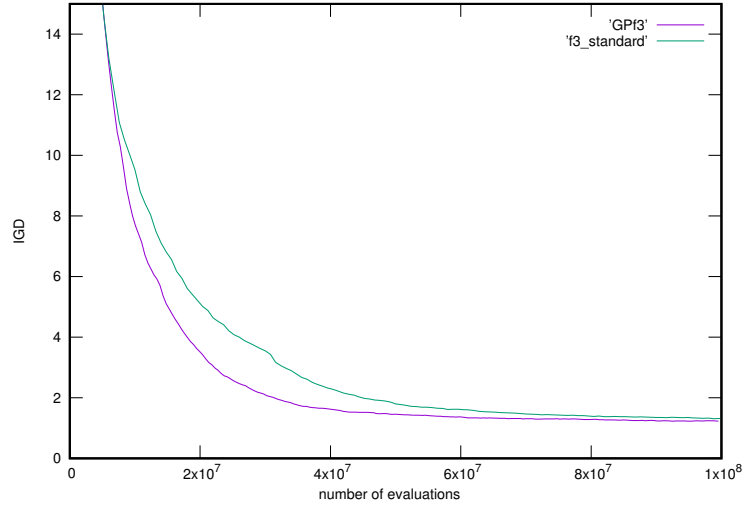


Figure 6.6: The convergence plot of surrogate-assisted point-based algorithm on TF3

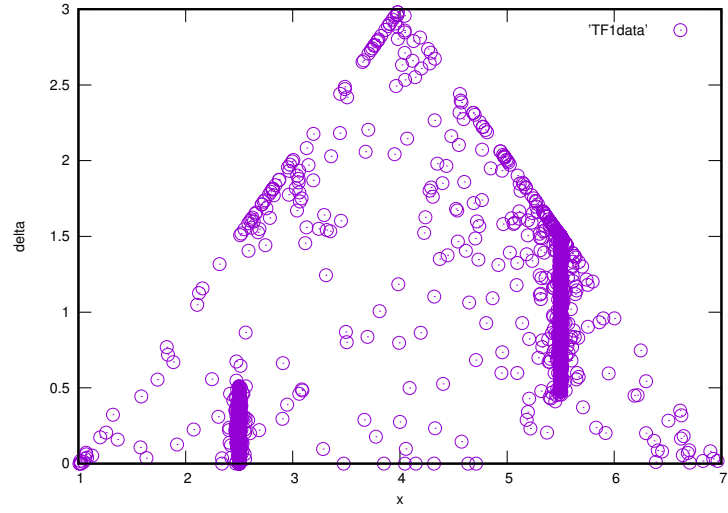


Figure 6.7: The stored solutions archive of TF1

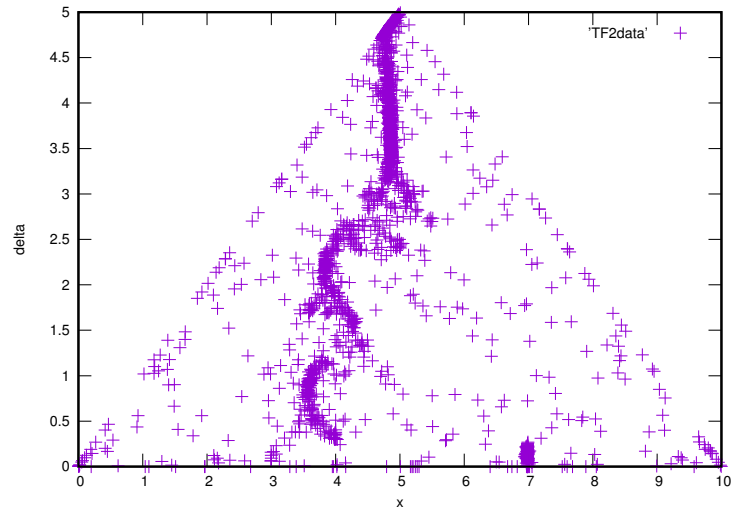


Figure 6.8: The stored solutions archive of TF2

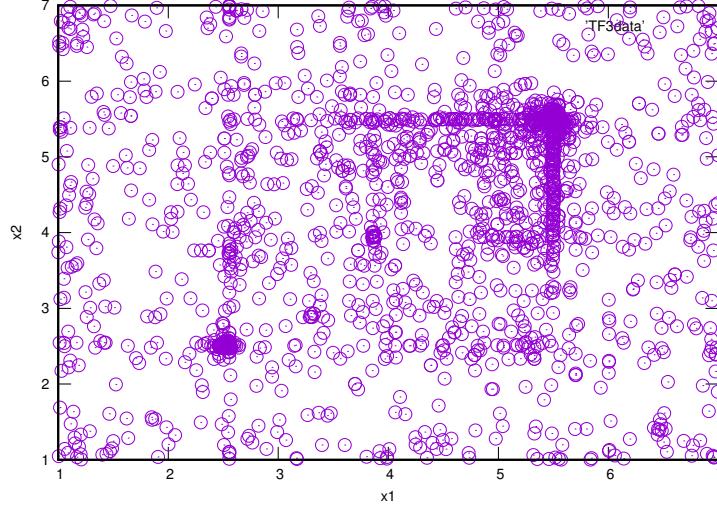


Figure 6.9: The stored solutions archive of TF3

from 0 to 0.5, $x_2 = 4.0$ with δ is 3.0, and $x_3 = 5.5$ with δ from 0.5 to 1.5. They are the optimal solutions of test function 1. Similarly in Figure 6.8, the solutions also evolve to the optimal distribution. For the 2-dimensional problem, the contour is shown in Figure 6.9 with optimal solutions in its highest and lowest peaks ($x_1 = 2.5, x_2 = 2.5, \delta \in [0, 0.5]$), ($x_1 = 5.5, x_2 = 5.5, \delta \in (0.5, 1.5)$), and ($x_1 = 4.0, x_2 = 4.0, \delta = 3.0$). The algorithm searches towards the optimal solutions. This demonstrates that by using GP, solutions could be guided to promising direction.

This surrogate-assisted algorithm can save fitness evaluations significantly by avoid evaluating bad solutions. In other words, it helps to reduce the number of calls to the lower level optimiser. It can be extended to general bi-level optimisation problems, where we apply the GP model to approximate the lower level objectives and find lower level optimal solutions that are then returned to the upper level.

6.5.2 Empirical evaluation of surrogate-assisted envelope-based algorithm

In this surrogate-assisted envelope-based algorithm. GP is expected to construct to approximate the lower level trade-off for each upper level solution. The parameter setting for standard envelope-based MOEA is described in Chapter 5. In the envelope-based algorithm, each upper level solution corresponds to a lower level Pareto front (envelope). The number of points of this envelope is determined by the lower level population size. From the parameter setting, the lower level population size is set as 100 for the test functions. As it describes in the surrogate-assisted

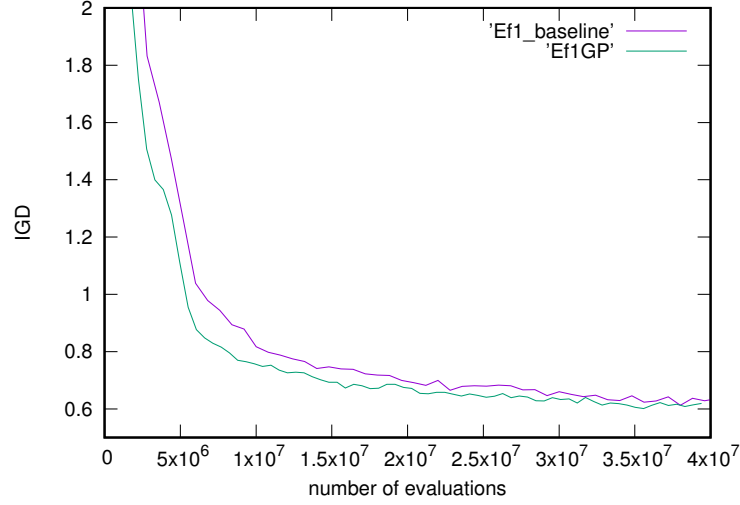


Figure 6.10: The convergence plot of surrogate-assisted envelope-based algorithm on TF1

envelope-based algorithm, two upper level solutions are selected. These two upper level solutions' lower level fronts consist of the data set that contains 200 points, and each point has an input (tolerance level) and output (worst-case fitness). Then we choose 100 points from this data set as the training data to build the GP model.

Test results and analysis of surrogate-assisted envelope-based algorithm

We compare the proposed surrogate-assisted envelope-based algorithm with the baseline envelope-based algorithm. The test results are based on three test functions. Figure 6.10, 6.11 and 6.12 shows the convergence plots. The purple convergence curves represent the baseline envelope-based algorithm, and the other line illustrates how IGD change with the number of evaluations. It is clear from all the three figures that the surrogate-assisted algorithm starts from a lower IGD value and converges faster than the baseline algorithm. In the beginning of the implementation, the available information is less, the fitness evaluation saving is less. As more evaluated upper level solution stored in the archive, the prediction becomes more accurate.

6.6 Summary

In this chapter, we propose the point-based and envelope-based surrogate-assisted algorithms to accelerate the baseline algorithms. The purpose is to build a surrogate model to approximate the worst-case fitness to identify the good upper level

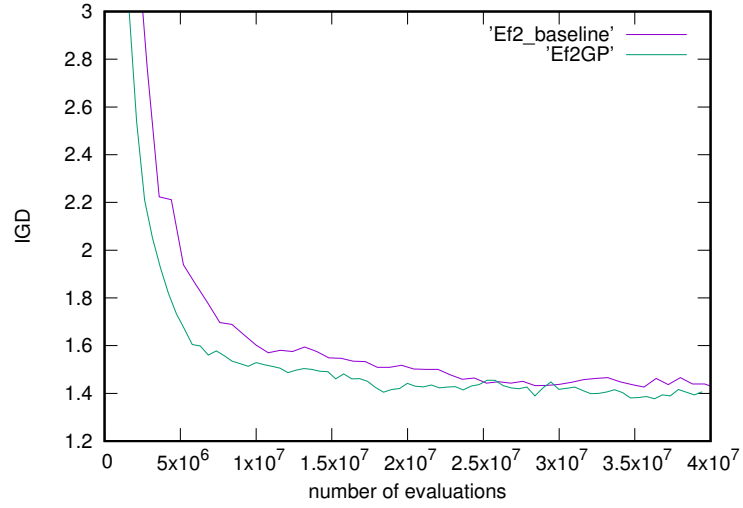


Figure 6.11: The convergence plot of surrogate-assisted envelope-based algorithm on TF2

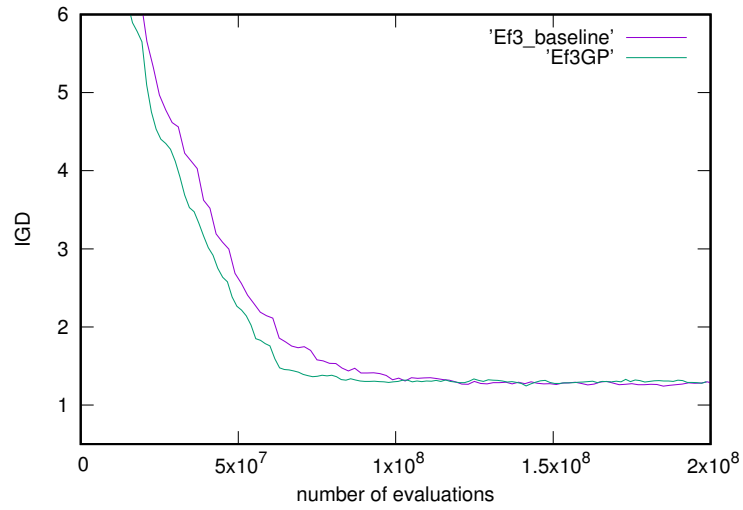


Figure 6.12: The convergence plot of surrogate-assisted envelope-based algorithm on TF3

solutions. Based on the estimation, only the promising upper level solutions are evaluated by calling the lower level optimiser. From the test results of the point-based surrogate-assisted algorithm, we can see that the algorithm searches towards the optimal solutions. The empirical results of the envelope-based surrogate-assisted algorithm shows that the algorithm with surrogate models outperforms the baseline algorithm.

If solution is expensive to evaluate, the computational effort will be significant in an optimisation problem. Usually, a surrogate model is constructed to approximate the fitness of a solution. Based on the estimation, those promising solutions can be filtered. Therefore only those promising solutions will be evaluated by its actual fitness. The computational cost can be saved by not evaluating bad solutions identified by surrogate model. GP is considered as a good surrogate model to approximate the fitness, because it allows to give an error bar about the approximation. It might take computational time to build a GP model for large data sets. If constructing a GP model is more expensive than evaluating a function, then there may have a balance between the computational cost of building a GP model and evaluating a function.

Chapter 7

Conclusion and future work

Optimisation problems with uncertainty have been discussed extensively. In practice, if a solution is very sensitive to small variations, then this solution may not be good to use. In this thesis, we considered optimisation problems with uncertainty in the decision space. Solutions that not only have good quality but also have tolerance against the uncertainty are preferred. For a solution disturbed by uncertainty, its worst-case quality in the neighbourhood is considered. We formulated the problem as a nested multi-objective optimisation problem. The worst-case quality and robustness are two objectives that need to be maximised in our problem definition. The aim is to search for the trade-off between robustness and worst-case quality. Two nested MOEAs are suggested to find the trade-off between the robustness and worst-case quality. Then, in order to improve the efficiency of both algorithms, we propose a few simple strategies to save fitness evaluations. Those strategies focus on modifying the lower level algorithms. Finally, we suggest two surrogate-assisted algorithms to accelerate the baseline algorithms.

7.1 Research summary

Firstly, the problem is formulated as a multi-objective optimisation problem where we examine the trade-off between worst-case quality and robustness. This trade-off allows the decision maker to know the solution with the best worst-case performance for any tolerance level. In order to solve this problem, two newly suggested multi-objective evolutionary algorithms are proposed. One is point-based MOEA where the lower level is a single-objective optimisation problem and returns a point to the upper level, and the other is envelope-based MOEA where the lower level is a multi-objective optimisation problem and returns a Pareto front (envelope) to the

upper level. Both can solve the problem well, and performance is measured by IGD. A lower IGD value indicates that the solutions obtained by our algorithms are closer to the true optimal solutions. Both algorithms are bi-level optimisation algorithms, where each feasible upper level solution is obtained by solving a lower level optimisation problem. This is computationally expensive especially when both levels are population-based algorithms that require a large number of evaluations. In order to improve the efficiency of those algorithms, we propose and compare several simple strategies to reduce the number of fitness evaluations.

Later, two surrogate-assisted algorithms are suggested to combine surrogate models with our multi-objective evolutionary algorithms. The surrogate models aim to identify promising upper level solutions that should be evaluated accurately by running the lower level optimiser fully. In a general bi-level optimisation problem, it always requires a large number of fitness evaluations because an upper level solution is evaluated by a lower level optimiser. The use of a surrogate model can help reduce the computational cost significantly. We propose to build a GP to predict the worst-case fitness of a solution and define a new criterion to determine whether to use the prediction as the fitness evaluation. This method can help identify which of those upper level solutions are potentially good and should be evaluated by calling the lower level optimiser. We combine this method with MOEAs to solve the proposed problem more efficiently. A number of continuous and discontinuous test functions are used to test our newly proposed framework, and the results are presented above. The total number of fitness evaluations could be reduced significantly without sacrificing the solution quality much, sometimes even allowing to find better solutions.

7.2 Contributions

The main contributions of this thesis are:

1. Problem formulation

We look at optimisation problems with uncertainty in the decision space, and search for robust solutions that not only have good quality, but also have tolerance against uncertainty. For a decision variable x with tolerance level δ , its worst-case performance is defined as $f^w(x, \delta) = \min f(x')$, where $x' \in [x - \delta, x + \delta]$. The robustness δ is defined as the maximum allowed deviation from the decision variables x . Our problem is formulated where the trade-off between robustness and worst-case quality is studied. For any specific

tolerance level, the solution with the best worst-case quality is identified. This research appears to be the first study to examine the trade-off between worst-case quality and robustness.

2. Point-based Multi-objective Evolutionary Algorithm

A newly developed point-based MOEA is suggested to solve the defined optimisation problem. This algorithm is defined as point-based because the lower level is a single objective optimisation problem. For an upper level solution (x_i, δ_i) , the lower level aims to search for its worst-case quality $f^w(x_i, \delta_i) = \min f(x')$, where $x' \in [x_i - \delta_i, x_i + \delta_i]$. In this point-based algorithm, the robustness for a decision variable x_i is set in the upper level.

3. Envelope-based Multi-objective Evolutionary Algorithm

The second proposed algorithm is an envelope-based MOEA, because the lower level is a multi-objective optimisation problem. The lower level will generate a Pareto front that is formed by a set of non-dominated solutions, and we call the lower level front an “envelope”. For an upper level solution x , its maximum robustness can be defined as Δ . The lower level decision variable x' is bounded to $[x - \Delta, x + \Delta]$, and its robustness is defined as the deviation from x' . The lower level attempts to search for a trade-off between the worst-case quality and robustness. In this envelope-based algorithm, the robustness for a solution is determined in the lower level. Because the lower level returns a Pareto front, the upper level population corresponds to a set of Pareto fronts. The upper level uses marginal hypervolume for selection, which is defined as the amount of decreased hypervolume if a specific upper level solution from the population would be removed. An upper level solution with larger marginal hypervolume is considered to be more important. The upper level Pareto front may contain the whole or partial lower level front.

4. Simple strategies to improve the efficiency of both algorithms

In bi-level optimisation problem, the lower level accuracy will influence the upper level search direction. It makes sense to give the lower level algorithm sufficient evaluations to improve the accuracy of the lower level. In some cases, a small number of evaluations will be sufficient to obtain a reasonably good lower level solution. Strategy I - VI will attempt to reduce the number of evaluation by modifying the lower level algorithm.

Both point-based and envelope-based algorithms are multi-objective evolutionary algorithms, and a larger number of fitness evaluations are required to

obtain optimal solutions. In order to improve the efficiency of our algorithms, we proposed several strategies to save fitness evaluations. Each strategy can save fitness evaluations to different levels, and up to 97% evaluations can be saved with a reasonable IGD value.

Strategy I exploits the upper level information for early abortion of the lower level. Strategy II exploits the information in the neighbourhood to improve the lower level solution quality. Strategy III also modifies the re-evaluation process at the end of each upper level generation. Strategy IV applies lower level smart initialisation and aims to save fitness evaluations in the re-evaluation process. Strategy V and VI make the lower level generation size or population size adaptive to the tolerance levels. It makes sense to use smaller number of generations or population size for a smaller search space. Strategy I works well across different test functions and it attempts to identify promising upper level solutions. The other five strategies focus on the lower level evaluations saving. Strategy I is combined with the other five strategies to improve the efficiency of both algorithms further.

5. Surrogate-assisted point-based algorithm

Compared to the proposed simple strategies that are combined with the baseline algorithms, we also propose to combine surrogate models with the baseline algorithm. This surrogate-assisted algorithm framework uses surrogate to filter the upper level population and only fully evaluates promising upper level solutions. In our problem, we apply GP to learn the robust function that is the worst-case quality f^w . Different from Strategy I this allows to identify the promising upper level solutions without running the lower level algorithm. This mechanism is called “pre-selection”. The surrogate-assisted point-based based algorithm guides the search towards the optimal solutions.

As for how to identify the promising upper level solutions based on the estimates, we compare the upper bound of each prediction with the dominated region. Because the upper level maximises the worst-case quality and robustness, the upper bound of the estimate needs to be focused. For a newly generated upper level solution, if its upper bound of the estimate enters into the dominated region, it is less likely to contribute to the upper level optimal solutions. If its upper bound of the estimate is above the dominated region, then it is more likely to contribute to the upper level optimal solutions and considered as promising in the upper level. In this case, it deserves to be evaluated accurately to check if it can improve the upper level Pareto front.

6. Surrogate-assisted envelope-based algorithm

In the surrogate-assisted point-based algorithm, for an upper level solution, the GP is used to approximate its worst-case quality. On the other hand, in the surrogate-assisted envelope-based algorithm, the surrogate model is used to approximate the lower level front. In our baseline envelope-based algorithm, the lower level is a multi-objective minimisation problem that returns a trade-off between worst-case quality and robustness. Since robustness is defined as the deviation from the decision variables, there is no need to evaluate. For an upper level solution, the GP is used to approximate its worst-case quality for a number of different tolerance levels.

As described in the surrogate-assisted point-based algorithm, the criterion of identifying the “good” upper level solutions is to check the upper bound of each prediction. In our surrogate-assisted envelope-based algorithm, we also check the upper bound of each prediction. Looking at the estimated lower level front, if there is any solution on the lower level front then its upper bound is above the dominated region, this solution could contribute to the upper level Pareto front. Therefore the corresponding upper level solution will be evaluated by calling the lower level optimiser.

7.3 Limitations

This study should be understood in light of its limitations.

In our problem framework, the lower level objective is the same as the upper level. With the regard of the algorithms and techniques used in this research, if applied to general bi-level optimisation problems then some ideas may need adaptation.

There are some parameters in the algorithms. How to set the parameters is challenging and depends on the problem. Our developed algorithms are based on EAs that involve parameters setting such as the population size, generation size, and crossover and mutation probability.

The two point-based and envelope-based algorithms are considered as bi-level algorithms. For a given number of evaluations, budget allocation to each level is crucial. The lower level accuracy will affect the upper level search direction. Because the lower level searches for the worst-case quality, while the upper level maximises the worst-case fitness. If the lower level algorithm can not identify the true worst-case quality for an upper level solution correctly, then it is possible that the lower level will identify a seemingly good solution. But the true worst-case fitness should

be worse than the obtained solution. If this seemingly good solution returns to the upper level, the upper level algorithm thinks this solution is a good one and keep it. In this case, the upper level search will be misguided. In order to improve the accuracy of the lower level, it is necessary to give lower level algorithm more evaluations. However, if we give the lower level algorithm more budget, the upper level may not have sufficient evaluations to find the optimal upper level solutions.

7.4 Future research work

There are many avenues for future work. Clearly the approaches should be tested on additional problems, including higher dimensional test problems or even a real-world problem.

Several strategies have been proposed to reduce the number of fitness evaluations in the point-based and envelope-based algorithms. There is room for further studies in how to improve the efficiency of bi-level worst-case optimisation. We would consider: 1) Vary the use of strategies over time. 2) The combination of three strategies. 3) Use quadratic programming for small tolerance level. The advantage of using quadratic programming is that it can find an exact solution if the problem is quadratic. In the point-based algorithm, the lower level is a single objective optimisation problem and it searches the worst-case quality in the neighbourhood. In a small enough neighbourhood, one can assume the problem is quadratic. For smaller tolerance level, the search space is relatively small and it is unlikely to contain local optima. If we replace the lower level EAs by quadratic programming, the problem becomes single level.

We suggested two surrogate-assisted algorithms to accelerate our baseline algorithms. In our problem, we use surrogate model to approximate the worst-case fitness. For each input design with its tolerance level, we could try to learn the expected value instead of the worst-case quality. This is to extend GP model to learn the expected fitness. Another possible future work could be that use a surrogate model to approximate the objective function f . This aims to replace the evaluation of f by using the approximated values. For instance, for a solution with its tolerance level, build a surrogate model to learn how f changes in the neighbourhood and find the worst-case fitness.

Our techniques can also be extended to general bi-level optimisation problems with lower level is single objective optimisation. In addition, we can extend our approach to the envelope-based algorithm with lower level multi-objective optimisation. In other words, for each upper level solution, a surrogate model would

be used to approximate the trade-off between lower level objectives and return the optimal lower level decision variables to the upper level.

In both surrogate-assisted point-based and envelope-based algorithms, the upper bound of the confidence interval is checked to determine whether to call the lower level optimiser. Because in the upper level worst-case quality and robustness are maximised, the upper bound of each prediction is defined as $mean + \alpha * sd$, where sd is the standard deviation. A possible future exploration could be learn the effects of varying this parameter α .

We are interested in other concepts of robustness in both single objective and multi-objective optimisation problems. For instance, more research on active robustness needs to be undertaken. This is a relatively new proposed concept, and we would like to consider active robustness in our problems. A possible idea is to include a parameter that relies on the decision variables, and then for a decision variable with a specific tolerance level, the included parameter could be adjusted to have a good worst-case quality.

A thorough analysis of various robustness concepts, how worst-case fitness is defined, and the algorithm for searching for robust solutions is important to conduct. In particular for optimisation problem with uncertain design inputs and environmental parameters. This will increase the complexity of the algorithms. Furthermore, in our envelope-based algorithm, we define the relationship between Pareto fronts. A further study about the dominance relationship between Pareto fronts, or worst-case robustness is necessary.

Bibliography

- Alicino, S. & Vasile, M. (2014), An evolutionary approach to the solution of multi-objective min-max problems in evidence-based robust optimization, *in* ‘IEEE Congress on Evolutionary Computation’, IEEE, pp. 1179–1186.
- Allmendinger, R., Emmerich, M., Hakanen, J., Jin, Y. & Rigoni, E. (2017), ‘Surrogate-assisted multicriteria optimization: Complexities, prospective solutions, and business case’, *Journal of Multi-Criteria Decision Analysis* **24**(1-2), 5–24.
- Angelo, J. S., Krempser, E. & Barbosa, H. J. (2014), Differential evolution assisted by a surrogate model for bilevel programming problems, *in* ‘IEEE Congress on Evolutionary Computation’, IEEE, pp. 1784–1791.
- Anthony, D. & Keane, A. (2003a), ‘Robust-optimal design of a lightweight space structure using a genetic algorithm’, *AIAA Journal* **41**(8), 1601–1604.
- Anthony, D. & Keane, A. (2003b), ‘Robust-optimal design of a lightweight space structure using a genetic algorithm’, *AIAA journal* **41**(8), 1601–1604.
- Avigad, G. & Branke, J. (2008), Embedded evolutionary multi-objective optimization for worst case robustness, *in* ‘Proceedings of the 10th Annual Conference on Genetic and Evolutionary Computation’, ACM, pp. 617–624.
- Avigad, G., Moshaiov, A. & Brauner, N. (2005), MOEA-based approach to delayed decisions for robust conceptual design, *in* ‘Applications of Evolutionary Computing’, Springer, pp. 584–589.
- Barrico, C. & Antunes, C. H. (2006), Robustness analysis in multi-objective optimization using a degree of robustness concept, *in* ‘IEEE Congress on Evolutionary Computation’, IEEE, pp. 1887–1892.
- Bertsimas, D., Brown, D. B. & Caramanis, C. (2011), ‘Theory and applications of robust optimization’, *SIAM Review* **53**(3), 464–501.

- Bertsimas, D., Nohadani, O. & Teo, K. M. (2010), ‘Robust optimization for unconstrained simulation-based problems’, *Operations Research* **58**(1), 161–178.
- Beyer, H.-G. (2000), ‘Evolutionary algorithms in noisy environments: theoretical issues and guidelines for practice’, *Computer Methods in Applied Mechanics and Engineering* **186**(2), 239–267.
- Beyer, H.-G. & Sendhoff, B. (2006), Evolution strategies for robust optimization, in ‘IEEE Congress on Evolutionary Computation’, IEEE, pp. 1346–1353.
- Beyer, H.-G. & Sendhoff, B. (2007), ‘Robust optimization—a comprehensive survey’, *Computer Methods in Applied Mechanics and Engineering* **196**(33), 3190–3218.
- Branke, J. (1998), Creating robust solutions by means of evolutionary algorithms, in ‘Parallel Problem Solving from Nature PPSN VI’, Springer, pp. 119–128.
- Branke, J. (2001), *Evolutionary Optimization in Dynamic Environments*, Kluwer Academic Publishers.
- Branke, J., Avigad, G. & Moshaiov, A. (2013), ‘Multi-objective worst case optimization by means of evolutionary algorithms’, *Technical Report* .
- Branke, J., Deb, K. & Miettinen, K. (2008), *Multiobjective optimization: Interactive and evolutionary approaches*, Vol. 5252, Springer Science & Business Media.
- Branke, J. & Lu, K. (2015), Finding the trade-off between robustness and worst-case quality, in ‘Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation’, ACM, pp. 623–630.
- Branke, J. & Rosenbusch, J. (2008), New approaches to coevolutionary worst-case optimization, in ‘Parallel Problem Solving from Nature PPSN X’, Springer, pp. 144–153.
- Branke, J., Scheckenbach, B., Stein, M., Deb, K. & Schmeck, H. (2009), ‘Portfolio optimization with an envelope-based multi-objective evolutionary algorithm’, *European Journal of Operational Research* **199**(3), 684–693.
- Branke, J. & Schmidt, C. (2005), ‘Faster convergence by means of fitness estimation’, *Soft Computing* **9**(1), 13–20.
- Chen, W., Unkelbach, J., Trofimov, A., Madden, T., Kooy, H., Bortfeld, T. & Craft, D. (2012), ‘Including robustness in multi-criteria optimization for intensity-modulated proton therapy’, *Physics in Medicine and Biology* **57**(3), 591.

- Coello, C. C., Lamont, G. B. & Van Veldhuizen, D. A. (2007), *Evolutionary Algorithms for Solving Multi-Objective Problems*, Springer.
- Colson, B., Marcotte, P. & Savard, G. (2007), ‘An overview of bilevel optimization’, *Annals of Operations Research* **153**(1), 235–256.
- Deb, K. & Gupta, H. (2005), Searching for robust pareto-optimal solutions in multi-objective optimization, in ‘In Proceedings of the third International Conference on Evolutionary Multi-Criterion Optimization’, Vol. 2005, pp. 150–164.
- Deb, K. & Gupta, H. (2006), ‘Introducing robustness in multi-objective optimization’, *Evolutionary Computation* **14**(4), 463–494.
- Deb, K., Gupta, S., Daum, D., Branke, J., Mall, A. K. & Padmanabhan, D. (2009), ‘Reliability-based optimization using evolutionary algorithms’, *IEEE Transactions on Evolutionary Computation* **13**(5), 1054–1074.
- Deb, K., Pratap, A., Agarwal, S. & Meyarivan, T. (2002), ‘A fast and elitist multi-objective genetic algorithm: NSGA-II’, *IEEE Transactions on Evolutionary Computation* **6**(2), 182–197.
- Deb, K. & Sinha, A. (2009), ‘An evolutionary approach for bilevel multi-objective problems’, *Cutting-Edge Research Topics on Multiple Criteria Decision Making, Communications in Computer and Information Science* **35**, 17–24.
- Du, X. & Chen, W. (2000), ‘Towards a better understanding of modeling feasibility robustness in engineering design’, *Journal of Mechanical Design* **122**(4), 385–394.
- Eiben, A. E., Smith, J. E. et al. (2003), *Introduction to evolutionary computing*, Vol. 53, Springer.
- Forouraghi, B. (2000), ‘A genetic algorithm for multiobjective robust design’, *Applied Intelligence* **12**(3), 151–161.
- Gaspar-Cunha, A. & Covas, J. (2006), Robustness using multi-objective evolutionary algorithms, in ‘Applications of Soft Computing’, Springer, pp. 353–362.
- Gaspar-Cunha, A. & Covas, J. A. (2008), ‘Robustness in multi-objective optimization using evolutionary algorithms’, *Computational Optimization and Applications* **39**(1), 75–96.
- Goh, C. K. & Tan, K. C. (2007a), Evolving the tradeoffs between pareto-optimality and robustness in multi-objective evolutionary algorithms, in ‘Evolutionary Computation in Dynamic and Uncertain Environments’, Springer, pp. 457–478.

- Goh, C. K. & Tan, K. C. (2007b), ‘An investigation on noisy environments in evolutionary multiobjective optimization’, *IEEE Transactions on Evolutionary Computation* **11**(3), 354–381.
- Greiner, H. (1996), ‘Robust optical coating design with evolutionary strategies’, *Applied Optics* **35**(28), 5477–5483.
- Gu, X., Sun, G., Li, G., Mao, L. & Li, Q. (2013), ‘A comparative study on multiobjective reliable and robust optimization for crashworthiness design of vehicle structure’, *Structural and Multidisciplinary Optimization* **48**(3), 669–684.
- Gunawan, S. & Azarm, S. (2005), ‘Multi-objective robust optimization using a sensitivity region concept’, *Structural and Multidisciplinary Optimization* **29**(1), 50–60.
- Gupta, H. & Deb, K. (2005), Handling constraints in robust multi-objective optimization, in ‘IEEE Congress on Evolutionary Computation’, Vol. 1, IEEE, pp. 25–32.
- Herrmann, J. W. (1999), A genetic algorithm for minimax optimization problems, in ‘IEEE Congress on Evolutionary Computation’, Vol. 2, IEEE.
- Ide, J. & Schöbel, A. (2016), ‘Robustness for uncertain multi-objective optimization: a survey and analysis of different concepts’, *OR Spectrum* **38**(1), 235–271.
- Ide, J., Tiedemann, M., Westphal, S. & Haiduk, F. (2015), ‘An application of deterministic and robust optimization in the wood cutting industry’, *4OR* **13**(1), 35–57.
- Jensen, M. T. (2001), Finding worst-case flexible schedules using coevolution, in ‘Proceedings of the Genetic and Evolutionary Computation Conference’, pp. 1144–1151.
- Jensen, M. T. (2004), A new look at solving minimax problems with coevolutionary genetic algorithms, in ‘Metaheuristics: Computer Decision Making’, Springer, pp. 369–384.
- Jin, Y. (2011), ‘Surrogate-assisted evolutionary computation: Recent advances and future challenges’, *Swarm and Evolutionary Computation* **1**(2), 61–70.
- Jin, Y. & Branke, J. (2005), ‘Evolutionary optimization in uncertain environments-a survey’, *IEEE Transactions on Evolutionary Computation* **9**(3), 303–317.
- Jin, Y. & Sendhoff, B. (2003), Trade-off between performance and robustness: an evolutionary multiobjective approach, in ‘Evolutionary Multi-Criterion Optimization’, Springer, pp. 237–251.

- Kalyanmoy Deb, A. S. (2010), ‘An efficient and accurate solution methodology for bilevel multi-objective programming problems using hybrid evolutionary-local-search algorithm’, *Evolutionary Computation* **18**(3), 403–449.
- Knowles, J. & Corne, D. (2002), On metrics for comparing nondominated sets, *in* ‘IEEE Congress on Evolutionary Computation’, Vol. 1, IEEE, pp. 711–716.
- Kruisselbrink, J., Emmerich, M. & Bäck, T. (2010), An archive maintenance scheme for finding robust solutions, *in* ‘International Conference on Parallel Problem Solving from Nature’, Springer, pp. 214–223.
- Kuroiwa, D. & Lee, G. M. (2012), ‘On robust multiobjective optimization’, *Vietnam Journal of Mathematics* **40**(2-3), 305–317.
- Lee, K.-H. & Park, G.-J. (2001), ‘Robust optimization considering tolerances of design variables’, *Computers and Structures* **79**(1), 77–86.
- Li, M., Azarm, S. & Aute, V. (2005), A multi-objective genetic algorithm for robust design optimization, *in* ‘The 7th Annual Conference on Genetic and Evolutionary Computation’, ACM, pp. 771–778.
- Lim, D., Ong, Y.-S., Jin, Y., Sendhoff, B. & Lee, B. S. (2007), ‘Inverse multi-objective robust evolutionary design’, *Genetic Programming and Evolvable Machines* **7**(4), 383–404.
- Lim, D., Ong, Y.-S. & Lee, B.-S. (2005), Inverse multi-objective robust evolutionary design optimization in the presence of uncertainty, *in* ‘The 7th Annual Conference on Genetic and Evolutionary Computation’, ACM, pp. 55–62.
- Lu, K., Branke, J. & Ray, T. (2016), Improving efficiency of bi-level worst case optimization, *in* ‘International Conference on Parallel Problem Solving from Nature’, Springer, pp. 410–420.
- Luo, B. & Zheng, J. (2008), A new methodology for searching robust pareto optimal solutions with moeas, *in* ‘IEEE Congress on Evolutionary Computation’, IEEE, pp. 580–586.
- Marzat, J., Walter, E. & Piet-Lahanier, H. (2013), ‘Worst-case global optimization of black-box functions through kriging and relaxation’, *Journal of Global Optimization* **55**(4), 707–727.
- Meneghini, I. R., Guimarães, F. G. & Gaspar-Cunha, A. (2016), Competitive co-evolutionary algorithm for robust multi-objective optimization: The worst case

minimization, in ‘IEEE Congress on Evolutionary Computation’, IEEE, pp. 586–593.

Mühlenbein, H. & Schlierkamp-Voosen, D. (1993), ‘Predictive models for the breeder genetic algorithm i. continuous parameter optimization’, *Evolutionary computation* **1**(1), 25–49.

Oduguwa, V. & Roy, R. (2002), Bi-level optimisation using genetic algorithm, in ‘IEEE International Conference on Artificial Intelligence Systems’, IEEE, pp. 322–327.

Ong, Y. S., Nair, P. B. & Keane, A. J. (2003), ‘Evolutionary optimization of computationally expensive problems via surrogate modeling’, *AIAA Journal* **41**(4), 687–696.

Ong, Y.-S., Nair, P. B. & Lum, K. Y. (2006), ‘Max-min surrogate-assisted evolutionary algorithm for robust design’, *IEEE Transactions on Evolutionary Computation* **10**(4), 392–404.

Paenke, I., Branke, J. & Jin, Y. (2006), ‘Efficient search for robust solutions by means of evolutionary algorithms and fitness approximation’, *IEEE Transactions on Evolutionary Computation* **10**(4), 405–420.

Rasmussen, C. E. & Williams, C. K. (2006), *Gaussian processes for machine learning*, Vol. 1, MIT press Cambridge.

Ray, T. (2002), Constrained robust optimal design using a multiobjective evolutionary algorithm, in ‘IEEE Congress on Evolutionary Computation’, Vol. 1, IEEE, pp. 419–424.

Ray, T. & Smith, W. (2006), ‘A surrogate assisted parallel multiobjective evolutionary algorithm for robust engineering design’, *Engineering Optimization* **38**(8), 997–1011.

Roy, B. (2010), ‘Robustness in operational research and decision aiding: A multifaceted issue’, *European Journal of Operational Research* **200**(3), 629–638.

Saha, A., Ray, T. & Smith, W. (2011), Towards practical evolutionary robust multi-objective optimization, in ‘IEEE Congress on Evolutionary Computation’, IEEE, pp. 2123–2130.

- Salomon, S., Avigad, G., Fleming, P. J. & Purshouse, R. C. (2013), Optimization of adaptation-a multi-objective approach for optimizing changes to design parameters, *in* ‘Evolutionary Multi-Criterion Optimization’, Springer, pp. 21–35.
- Salomon, S., Avigad, G., Fleming, P. J. & Purshouse, R. C. (2014), ‘Active robust optimization: enhancing robustness to uncertain environments’, *IEEE Transactions on Cybernetics* **44**(11), 2221–2231.
- Salomon, S., Purshouse, R. C., Avigad, G. & Fleming, P. J. (2015), An evolutionary approach to active robust multiobjective optimisation, *in* ‘Evolutionary Multi-Criterion Optimization’, Springer, pp. 141–155.
- Sebald, A. V. & Schlenzig, J. (1994), ‘Minimax design of neural net controllers for highly uncertain plants’, *IEEE Transactions on Neural Networks* **5**(1), 73–82.
- Sinha, A. (2011), Bilevel multi-objective optimization problem solving using progressively interactive emo, *in* ‘International Conference on Evolutionary Multi-Criterion Optimization’, Springer, pp. 269–284.
- Sinha, A., Malo, P. & Deb, K. (2013), ‘Efficient evolutionary algorithm for single-objective bilevel optimization’, *arXiv preprint arXiv:1303.3901* .
- Sinha, A., Malo, P. & Deb, K. (2014a), An improved bilevel evolutionary algorithm based on quadratic approximations, *in* ‘IEEE Congress on Evolutionary Computation’, IEEE, pp. 1870–1877.
- Sinha, A., Malo, P. & Deb, K. (2014b), ‘Test problem construction for single-objective bilevel optimization’, *Evolutionary Computation* **22**(3), 439–477.
- Sinha, A., Malo, P., Deb, K., Korhonen, P. & Wallenius, J. (2016), ‘Solving bilevel multicriterion optimization problems with lower level decision uncertainty’, *IEEE Transactions on Evolutionary Computation* **20**(2), 199–217.
- Tabatabaei, M., Hakanen, J., Hartikainen, M., Miettinen, K. & Sindhya, K. (2015), ‘A survey on handling computationally expensive multiobjective optimization problems using surrogates: non-nature inspired methods’, *Structural and Multidisciplinary Optimization* **52**(1), 1–25.
- Thompson, A. (1998), On the automatic design of robust electronics through artificial evolution, *in* ‘Evolvable Systems: From Biology to Hardware’, Springer, pp. 13–24.

- Tsutsui, S. (1999), A comparative study on the effects of adding perturbations to phenotypic parameters in genetic algorithms with a robust solution searching scheme, *in* ‘IEEE International Conference on Systems, Man, and Cybernetics’, Vol. 3, IEEE, pp. 585–591.
- Tsutsui, S. & Ghosh, A. (1997), ‘Genetic algorithms with a robust solution searching scheme’, *IEEE Transactions on Evolutionary Computation* **1**(3), 201–208.
- Tsutsui, S. & Ghosh, A. (2003), Effects of adding perturbations to phenotypic parameters in genetic algorithms for searching robust solutions, *in* ‘Advances in Evolutionary Computing’, Springer, pp. 351–365.
- Tsutsui, S., Ghosh, A. & Fujimoto, Y. (1996), A robust solution searching scheme in genetic search, *in* ‘Parallel Problem Solving from Nature PPSN IV’, Springer, pp. 543–552.
- Vasile, M. (2014), On the solution of min-max problems in robust optimization, *in* ‘The EVOLVE 2014 International Conference, A Bridge between Probability, Set Oriented Numerics, and Evolutionary Computing’.
- Wiesmann, D., Hammel, U. & Bäck, T. (1998), ‘Robust design of multilayer optical coatings by means of evolutionary algorithms’, *IEEE Transactions on Evolutionary Computation* **2**(4), 162–167.
- Zhou, A. & Zhang, Q. (2010), A surrogate-assisted evolutionary algorithm for min-max optimization, *in* ‘IEEE Congress on Evolutionary Computation’, IEEE, pp. 1–7.
- Zhou, Z., Ong, Y. S. & Nair, P. B. (2004), Hierarchical surrogate-assisted evolutionary optimization framework, *in* ‘IEEE Congress on Evolutionary Computation’, Vol. 2, IEEE, pp. 1586–1593.
- Zhou, Z., Ong, Y. S., Nair, P. B., Keane, A. J. & Lum, K. Y. (2007), ‘Combining global and local surrogate models to accelerate evolutionary optimization’, *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* **37**(1), 66–76.
- Zitzler, E. & Thiele, L. (1998), Multiobjective optimization using evolutionary algorithms—a comparative case study, *in* ‘Parallel Problem Solving from Nature PPSN V’, Springer, pp. 292–301.

Zitzler, E., Thiele, L., Laumanns, M., Fonseca, C. M. & Da Fonseca, V. G. (2003),
‘Performance assessment of multiobjective optimizers: an analysis and review’,
IEEE Transactions on Evolutionary Computation **7**(2), 117–132.